VMS Networking Manual

Order Number: AA–LA48A–TE

April 1988

This book presents conceptual and usage information for VMS users who want to manage DECnet–VAX, perform operations over the network, or both.

Revision/Update Information:	This manual supersedes the *VAX/VMS Networking Manual*, Version 4.4.
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

ZK4520

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.
In New Hampshire, Alaska, and Hawaii call 603-884-6660.
In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).
Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

TABLES

Preface

The *VMS Networking Manual* presents an introduction to networking software used on VMS operating systems. It provides a conceptual description of DECnet–VAX software used to access the DECnet network, and VAX Packetnet System Interface (PSI) software used to access packet switching data networks. This manual explains how to configure and manage the network using the VMS Network Control Program (NCP), the primary tool for network management. It also explains how to perform user operations over the network.

Intended Audience

The *VMS Networking Manual* is intended for those who perform network management functions to control, monitor, or test DECnet–VAX and VAX PSI software running on a VMS operating system. This manual is also intended for VMS users who perform remote file access or task-to-task operations using DECnet–VAX. You are assumed to be familiar with the VMS operating system, but not necessarily experienced with DECnet operations.

Document Structure

The *VMS Networking Manual* is divided into four major parts:

- Part I introduces you to basic networking concepts required to understand DECnet–VAX operations, and indicates how you can interact with the network.
- Part II provides usage information to those responsible for DECnet–VAX system management, and explains how to use the Network Control Program to manage the network and perform VMS host services to remote systems (such as downline loading and upline dumping).
- Part III specifies the procedures for configuring, installing, and testing DECnet–VAX and VAX PSI on a VMS operating system.
- Part IV describes the techniques for carrying out user operations over the network, including accessing remote files and performing task-to-task communications.

Associated Documents

The networking concepts and operations described in the *VMS Networking Manual* are directly related to the following four manuals:

VMS Mini-Reference

Provides a quick-reference summary of NCP command formats.

Guide to DECnet–VAX Networking

Provides a conceptual overview of networking concepts and DECnet–VAX. Also describes procedures for asynchronous communication.

VMS Network Control Program Manual

Provides usage information for the Network Control Program (NCP) Utility.

VMS DECnet Test Sender/DECnet Test Receiver Utility Manual

Provides usage information for the DECnet Test Sender/Receiver (DTS/DTR) Utility.

The *VMS License Management Utility Manual* describes the License Management Utility (LICENSE), which is used to enable product licenses, including the DECnet–VAX licenses.

The information in the *VMS Networking Manual* is also related to these other VMS manuals:

Overview of VMS Documentation

Describes the VMS documentation set.

VMS DCL Concepts Manual

Provides a conceptual overview of DCL, including the format of file specifications and the use of command procedures.

VMS DCL Dictionary

Describes all DCL commands, including SET HOST and SHOW NETWORK.

Guide to Using VMS Command Procedures

Describes the design, construction, and execution of command procedures.

VMS System Messages and Recovery Procedures Reference Volume

Explains all VMS messages, including messages issued by DECnet–VAX and by system services associated with network-related operations.

VMS VAXcluster Manual

Describes procedures for setting up and managing a VAXcluster.

Guide to Setting Up a VMS System

Describes system management procedures for setting up a system, including the procedures to create and use directories.

Guide to Maintaining a VMS System

Describes system management procedures for maintaining a system, including the use of virtual terminals, the control of user privileges, and the use of SYSGEN parameters.

VMS Record Management Services Manual

Describes Record Management Services (RMS) fields and options that are applicable to DECnet–VAX operations.

VMS I/O User's Reference Volume

Describes input/output operations, including the procedures for sending messages to an Ethernet multicast address.

Guide to VMS File Applications

Provides examples of MACRO programs for remote file access.

Guide to VMS Programming Resources

Provides general information about $QIO system services.

VMS Device Support Manual

Provides guidelines for elevated IPL programming.

VMS System Services Reference Manual

Describes system mailboxes, AST routines, and $QIO system services.

VMS Run-Time Library Routines Volume

Describes VMS Run-Time Library (RTL) routines, including routines for creating temporary mailboxes.

VMS Authorize Utility Manual

Describes how to use the Authorize Utility, including how to establish proxy login accounts.

Guide to VMS System Security

Describes system security guidelines, including how to establish proxy login accounts.

VMS System Generation Utility Manual

Describes SYSGEN procedures.

See also the *VMS Version 5.0 Release Notes*.

The *Introduction to DECnet Phase IV* manual, not part of the VMS documentation set, provides an overview of DECnet software. The *Routing and Networking Overview*, also not part of the VMS documentation set, provides an overview of DECnet routing.

For information about VAX PSI, refer to the following manuals, which make up the VAX PSI documentation set:

P.S.I. Introduction
VAX P.S.I. Installation Procedures
VAX P.S.I. X.25 Programmer's Guide
VAX P.S.I. X.29 Programmer's Guide
VAX P.S.I. Management Guide
VAX P.S.I. PAD and MAIL Utilities Manual
VAX P.S.I. Problem Solving Guide
Public Network Information

The following functional specifications define DIGITAL Network Architecture (DNA) protocols to which all implementations of DECnet adhere:

DECnet DIGITAL Network Architecture General Description
DIGITAL Data Communications Message Protocol Functional Specification
Network Services Protocol Functional Specification
Maintenance Operation Protocol Functional Specification
Data Access Protocol Functional Specification
Routing Layer Functional Specification
DNA Session Control Functional Specification
DNA Phase IV Network Management Functional Specification
Ethernet Node Product Architecture Specification
Ethernet Data Link Functional Specification

Conventions

Convention	Meaning
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

New and Changed Features

The VMS Version 5.0 technical changes to DECnet–VAX software can be grouped according to new features, enhanced procedures and components, and miscellaneous documentation changes.

The following features have been added:

- Support of the asterisk (*) and the percent sign (%) as wildcard characters to represent component names in NCP commands.
- Support for command line recall on the NCP command line. By pressing CTRL/B or the arrow keys you can recall multiple commands previously entered.
- NCP executor parameters to support equal cost path load splitting. If multiple paths to a destination node are equal in cost, a packet load can be split for routing over the multiple paths. The executor parameter MAXIMUM PATH SPLITS defines the number of equal cost paths among which the packet load is to be split. The executor parameter PATH SPLIT POLICY specifies whether a packet load is split equally over all equal cost paths.
- NCP line parameter HOLDBACK TIMER to control the maximum time for delay acknowledgments.
- Three new node commands that implement downline load support. The parameters LOAD ASSIST AGENT and LOAD ASSIST PARAMETER extend Ethernet downline load support to Local Area VAXcluster nodes. The MANAGEMENT FILE parameter identifies a file to be loaded downline to an adjacent node.
- Support for these circuit and line devices: the DELQA, DMB32, DEBNA, DESVA, DHQ11, and DZQ11.

The following procedures and components have been modified:

- NSP software includes support for out-of-order packet caching. This mechanism ensures delivery of packets even if they appear out of order due to equal cost path splitting.
- Ethernet end node caching includes support for reverse path caching, which improves routing between nodes that are not on the Ethernet, but can be accessed by a node on the Ethernet.
- The proxy database is now kept in the file NETPROXY.DAT, rather than in NETUAF.DAT.
- The executor parameter DEFAULT PROXY has been replaced by the parameters INCOMING PROXY and OUTGOING PROXY. A new NCP command, SET KNOWN PROXIES ALL, rebuilds the volatile proxy database from the contents of the permanent database.
- The target-initiated load/dump procedure has been modified so that NETACP will not start up the Maintenance Operation Module (MOM) process until it confirms that it has all the required information to start up the operation.

- The directory location of MOM load images and dump file has been changed. Previously, the default directory for these MOM images was in SYS$SYSTEM. The MOM load images and dump files are now located in their own private directory, MOM$SYSTEM.
- Some of the user prompts have been changed in the network configuration procedure, NETCONFIG.COM.
- The procedure for enabling the DECnet–VAX license has been modified. Previously, a specific DECnet–VAX license (either an end node license or a full function license) was enabled by installing the appropriate key distributed separately on magnetic media. The new License Management Utility is now used to register all keys. You must use the License Management Utility to register a DECnet–VAX key in order to enable the DECnet–VAX license on your system.
- The command procedure SYS$MANAGER:SYSTARTUP.COM has been replaced by SYS$MANAGER:SYSTARTUP_V5.COM.

The following miscellaneous documentation changes have been made:

- Extensive revisions to Chapter 8, including the addition of a new programming example for nontransparent task-to-task communication.
- Addition of descriptions of the following remote file operations:

  VMS to MS–DOS
  VMS to Ultrix
  VMS to MVS

- Removal of MicroVMS as a distinct operating system. As of Version 5.0, the VMS operating system running on MicroVAX machines is called VMS, not MicroVMS.

Part I Introduction to DECnet–VAX and VAX PSI

1 Overview of DECnet–VAX and VAX PSI

This chapter presents an overview of the networking software used on VMS systems: what the software is, how to manage it, and how to interface with it. This chapter introduces DECnet–VAX software, which enables access to the DECnet network, and VAX PSI software, which provides access to a packet switching network. You use the same VMS network management tools to manage both DECnet–VAX and VAX PSI.

The following sections introduce the network terms and concepts used throughout this manual, identify network software, describe network configurations, and provide a brief summary of network management responsibilities. The chapter also defines the application user's relationship to the network.

For details about specific topics, you should consult the pertinent chapters of this manual, other manuals in the VMS document set, and VAX PSI manuals (see the Preface).

1.1 General Description of a DECnet Network

Computer processes communicate with one another over a data network. This network consists of two or more computer systems called **nodes** and the **logical links** between them. A logical link is a connection, at the user level, between two processes. **Adjacent nodes** are connected by physical **lines** over which **circuits** operate. A circuit is a communications data path over which all input and output (I/O) between nodes takes place. A circuit can support many concurrent logical links.

Nodes can also be physically connected to a **packet switching data network (PSDN)** to allow DECnet circuits to be mapped to PSDN **virtual circuits**. Virtual circuits are logical associations between nodes for the exchange of data; the actual circuit employed is invisible to you. Alternatively, you can attach computers or terminals directly to a PSDN without using a DECnet data link, or use a connector node as a gateway to communicate with remote nodes over a PSDN.

In a network of more than two nodes, the process of directing a data message from a source to a destination node is called **routing**. DECnet supports adaptive routing, which permits messages to be routed through the network over the most cost-effective path; messages are rerouted automatically if a circuit becomes disabled.

Nodes can be either **routing nodes** (called **routers**) or **nonrouting nodes** (known as **end nodes**). Both routing nodes and end nodes can send messages to and receive messages from other nodes in the network. However, routing nodes have the ability to forward or route messages from one node to another when the two nodes exchanging these messages have no direct physical link between them, except for the path that includes the node forwarding the message. End nodes can never have more than one circuit connecting them with the network. Any node that has two or more circuits connecting it to the network must be a router.

Phase IV DECnet supports the configuration of very large, as well as small, networks. In a network that is not divided into multiple areas, a maximum of 1023 nodes is possible, but the optimum number of nodes is much less (approximately 200 to 300 nodes, depending on the topology). **Area routing** techniques permit configuration of very large networks, consisting of up to 63 areas, each containing a maximum of 1023 nodes. In a multiple-area network, nodes are grouped into separate **areas**, each functioning as a subnetwork. DECnet supports routing within each area and a second, higher level of routing that links the areas. Nodes that perform routing within a single area are referred to as **level 1 routers**; those that perform routing between areas as well as within their own area are called **level 2 routers** (or **area routers**).

1.2 DECnet–VAX and VAX PSI

You can configure DECnet–VAX networking software on all VMS operating systems. You can install and configure VAX PSI software on VMS operating systems. These software products are identified and described in the following subsections.

1.2.1 DECnet Interface with the VMS Operating System

DECnet is the collective name for the software and hardware products that are a means for various DIGITAL operating systems to participate in a network. DECnet–VAX is the implementation of DECnet that causes a VMS operating system to function as a network node. As the VMS network interface, DECnet–VAX supports both the **protocols** necessary for communicating over the network and the functions necessary for configuring, controlling, and monitoring the network. A DECnet–VAX node can communicate with other DECnet–VAX nodes in the network or with any other DIGITAL operating system that supports DECnet.

A DECnet multinode network is decentralized; that is, many nodes connected to the network can communicate with each other without having to go through a central node. As a member of a multinode network, your node can communicate with any other network node, not merely the nodes that reside next to you, and gain access to software facilities that may not exist on your local node. An advantage of this type of network is that it allows different applications running on separate nodes to share the facilities of any other node.

Optionally, very large DECnet networks can be divided into multiple areas, for the purpose of hierarchical (area) routing. Area routing introduces a second, higher level of routing between areas (groups of nodes), which results in less routing traffic throughout the network. Each node in a multiple-area network can still communicate with all other nodes in the network.

1.2.2 VAX Packetnet System Interface

VAX Packetnet System Interface (PSI) is a software product that allows the VMS user to communicate across PSDNs. VAX PSI implements the **CCITT X.25** and **X.29** recommendations (described in Section 1.3.4), and International Standards 7776 and 8208, providing a user interface to a PSDN. A PSDN consists of switching nodes connected by high-speed links, to which computers or terminals can be attached.

You can use VAX PSI to do the following:

- Link DECnet nodes across a PSDN through **data link mapping (DLM)**. This permits an X.25 virtual circuit to be used as a DECnet data link.
- Communicate directly across one or more PSDNs to a DIGITAL or non-DIGITAL computer over an X.25 virtual circuit.
- Communicate directly across one or more PSDNs to a character-mode terminal connected to a **packet assembly/disassembly (PAD)** device. The PAD may be privately owned or located within the PSDN.
- Communicate by way of one or more PSDNs between terminals on a local VAX PSI node and remote DIGITAL or non-DIGITAL computers. VAX PSI contains a host-based PAD that provides this capability.
- Communicate directly with another DIGITAL or non-DIGITAL computer without an intervening PSDN, provided the other computer implements International Standards 7776 and 8208. In this case, one computer acts as the **data terminal equipment (DTE)**, the other as the **data circuit-terminating equipment (DCE)**.

You can use an alternative version of VAX PSI, called VAX PSI Access, on a VMS node that does not connect directly to a PSDN. VAX PSI Access provides all the capabilities of VAX PSI. However, VAX PSI uses DECnet to connect to a **connector node**, which in turn connects to a PSDN. The VAX PSI Access node is known as a **host node**. The connector node may be a VAX PSI node in multihost mode or an X25router. Each connector node can connect to one or more PSDNs, and each host node can connect to one or more connector nodes.

You can install and configure both VAX PSI and VAX PSI Access on the same node, which is then known as a **combination node**. You may need a combination node if you want to connect directly to a PSDN in native or multihost mode and also want to have access to another PSDN by way of a connector node.

1.2.3 DECnet Functions

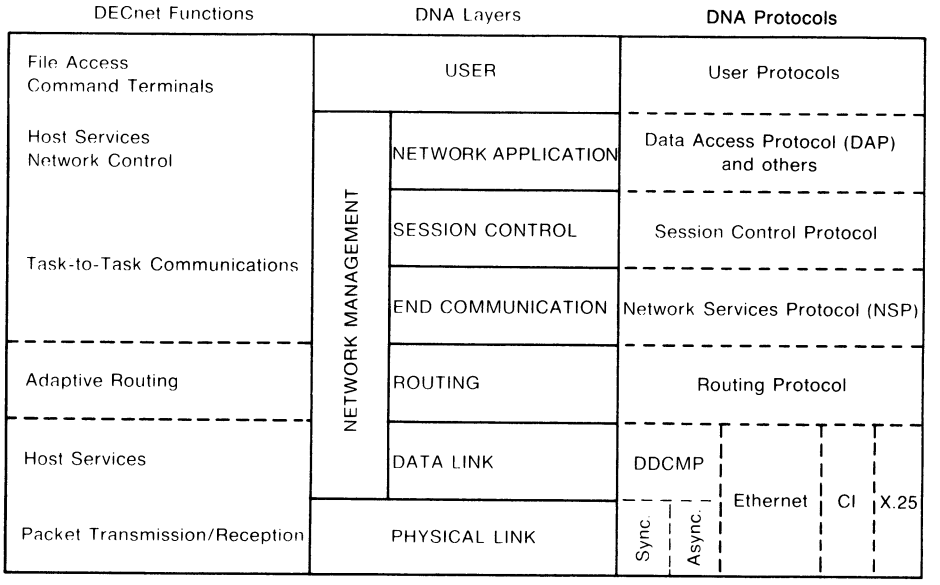
Networking functions you can perform using DECnet–VAX are as follows. These functions are introduced in this section and described in detail in later chapters, as indicated.

- Network management functions
  - Controlling the network (Chapters 2 through 7)
  - Providing DECnet–VAX host services to other DECnet nodes (Chapter 4)

  - Performing routing configuration and control (Chapters 2 and 3)
  - Establishing DECnet–VAX configurations (Chapters 2, 3, and 5)
- Applications user functions
  - Accessing files across the network (Chapters 8 and 9)
  - Using a heterogeneous command terminal (Chapter 8)
  - Performing task-to-task communications across the network (Chapter 8)

DECnet products are based on the layered network design specified in the DIGITAL Network Architecture (DNA). Figure 1–1 illustrates the DECnet functions, the various DNA layers at which they are initiated, and the DNA protocols by which these functions are implemented. Each DNA layer is a client of the next lower layer and does not function independently. For a complete description of DNA, see the DNA specifications. The DECnet–VAX configurations that use the Ethernet, DDCMP, CI, and X.25 protocols are defined in the following section.

Figure 1–1 DECnet Functions and Related DNA Layers and Protocols



ZK-1850-84

1.3 DECnet–VAX Configurations

DECnet supports network connections to the following:

- An Ethernet circuit in a local area network configuration
- A node running DECnet using the DIGITAL Data Communications Message Protocol (DDCMP): either a **synchronous** point-to-point or multipoint connection, or an **asynchronous** static or dynamic point-to-point connection
- Another node running DECnet over the computer interconnect (CI)
- A node running DECnet in a direct X.25 connection over a PSDN

Figure 1–2 illustrates a sample DECnet–VAX Phase IV configuration showing various kinds of DECnet–VAX nodes (VMS routers and end nodes, a VMS router in a VAXcluster, and VMS end nodes in a Local Area VAXcluster) connected to an Ethernet, and two Ethernets connected by means of routers. Figure 1–2 shows the use of a DDCMP synchronous line to connect a router to additional DECnet nodes, and static (permanent) asynchronous DDCMP lines to connect a router to VMS end nodes installed on VAXstations. It also indicates a dialup connection between a MicroVAX-based VMS system and a routing node in the VAXcluster, by means of a dynamic asynchronous DDCMP line (switched on for the length of the call).

Figure 1–2 demonstrates two kinds of connections to a PSDN: a VMS node connected directly to a PSDN by means of an X.25 circuit, and a VMS host node that can be connected to a PSDN by means of connector nodes that serve as gateways to the PSDN. One connector node is a VMS node running multihost PSI and the other is an X25router.

DECnet–VAX connections are described in the following subsections. A detailed discussion of the various types of circuits and lines used in a DECnet network is presented in Chapter 2.

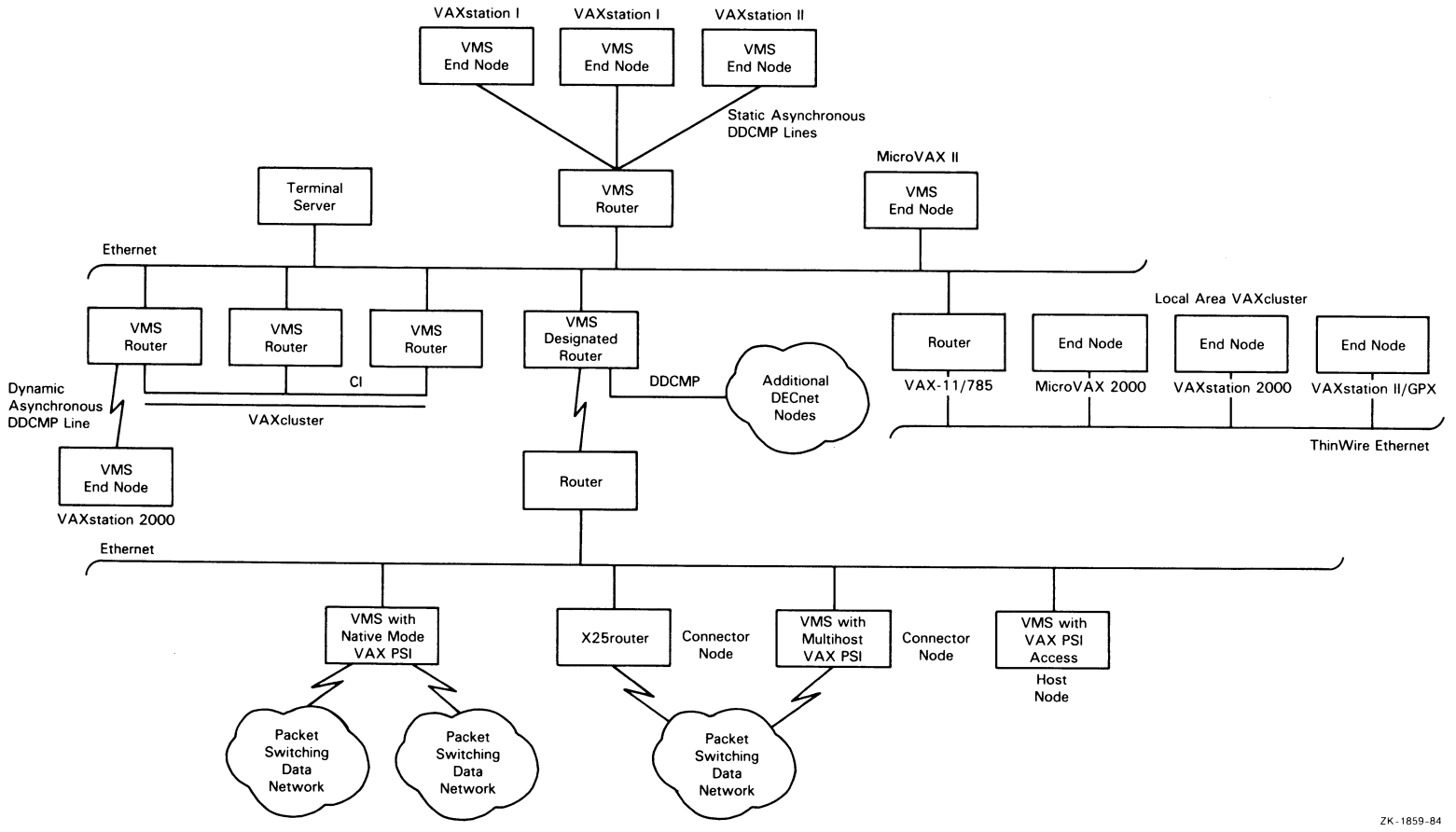
1.3.1 DECnet–VAX Ethernet Local Area Network Configuration

The Ethernet is a local area network component that provides a reliable high-speed communications **channel**, optimized to connect information processing equipment in a limited geographic area, such as an office, a building, or a complex of buildings (for example, a campus).

Local area networks (LANs) are designed for a wide variety of technologies and arranged in many configurations. Digital Equipment Corporation, Intel Corporation, and Xerox Corporation collaborated in producing the Ethernet specification to develop a variety of LAN products. DIGITAL's implementation of the Ethernet specification that was originated by the Xerox Corporation appears at the lowest two levels of the overall DNA specification: the Physical layer and the Data Link layer.

At the Physical layer, the Ethernet topology is a bus, in the shape of a branching tree, and the medium is a shielded coaxial cable that uses Manchester-encoded, digital baseband signaling. The maximum data rate is 10 million bits per second. Maximum use of an Ethernet's data transmission capability occurs when multiple pairs of nodes communicate simultaneously.

Figure 1–2 Sample DECnet–VAX Phase IV Configuration



In practice, DECnet transmission between a pair of nodes on an Ethernet occurs at a considerably lower rate. Each Ethernet can support up to 1023 nodes; the maximum possible distance between nodes on the Ethernet is 2.8 kilometers (1.74 miles).

At the Data Link layer, network control for the Ethernet is multiaccess, fairly distributed to all nodes. Ethernet access control is **CSMA/CD (Carrier Sense, Multiple Access with Collision Detect)**. The frame length allocation is from 64 to 1518 bytes (including an 18-byte envelope).

Section 2.2.4 lists the Ethernet circuit devices supported by DECnet–VAX.

1.3.1.1 Ethernet Datagrams

Message **packets** sent over Ethernet are called **datagrams**. Because there is no guarantee that a datagram will be received by the intended destinations, reliable connections (in the form of virtual circuits) may be provided by a protocol being interposed between the user and the Ethernet datagram service. In DNA, this virtual circuit is provided by the Network Services Protocol (NSP) in the End Communication layer.

Initialization of nodes on Ethernet is based on multicast addressing and the use of datagrams. It differs from initialization of nodes on DDCMP circuits in that it does not involve guaranteed delivery of routing messages.

1.3.1.2 Transmission and Reception of Ethernet Packets

An Ethernet is a single shared network channel, with many nodes demanding equal access to it. The technique used to mediate these demands is CSMA /CD. A good analogy for this technique is the interaction of people at a social gathering. To be polite, one does not speak while someone else is talking; that is, one listens before speaking. On the Ethernet, listening to determine whether the communication medium is already in use is called **carrier sense**. Messages are said to be *initially deferred* if they are not sent on the Ethernet because a transmission is in progress.

At a social gathering, anyone may begin to talk once he or she determines that no one else is; the ability of any station on the Ethernet to use the communication medium is known as **multiaccess**. If two or more people, detecting silence, start to talk at about the same time, they note the fact and stop talking (that is, each listens while talking and stops if interfering with someone else); the noting of the fact that more than one station is transmitting, followed by the cessation of communication, is called **collision detect**.

When two or more people at a social event start talking simultaneously, they stop talking, wait some random time, and start talking again; on an Ethernet, this situation is known as *backoff and retransmission*, and it is expected that a random delay before retransmission eventually clears the collision situation.

There is a further useful analogy between Ethernet and a social event. When one is talking to a group of people, everyone can hear everything said. Some of what is said is intended for everyone, some is intended for a subset of the group (say, everyone over 21), and some is intended for an individual. Stations on an Ethernet can hear every message. Some messages are intended for all stations (**broadcast address**), some are intended for a subset (**multicast address**), and some are intended for individual stations (**physical address**).

On an Ethernet, every station can listen to every message, and messages can be addressed to their intended recipient(s). These two features greatly increase the communications efficiency of a network that uses Ethernet over that of a completely connected DDCMP network.

1.3.1.3 Ethernet Routers and End Nodes

Ethernet supports connections to routers and end nodes. On an Ethernet, a routing node selected as a **designated router** can perform routing services on behalf of end nodes. In addition, routers can route packets between Ethernet nodes and non-Ethernet nodes (such as nodes on DDCMP circuits). An end node on an Ethernet can communicate directly with any other node (router or end node) on the same Ethernet by sending a message directly to the addressed node. Note that an end node on a non-Ethernet circuit can communicate only with an adjacent node on the same circuit.

1.3.2 DDCMP Network Configurations

DDCMP provides a low-level communications path between systems. The DDCMP protocol performs the basic communications function of moving information blocks over an unreliable communication channel. (The protocol detects any bit errors introduced by the channel and requests retransmission of the block.) You also use DDCMP to manage the orderly transmission and reception of blocks on channels with one or more transmitters and receivers.

The DDCMP protocol is supported on synchronous and asynchronous communications devices. DDCMP connections can be point-to-point or multipoint configurations. Point-to-point connections are either synchronous or asynchronous. The two types of asynchronous connections are static (permanent) and dynamic (switched temporary). Multipoint connections are always synchronous. These connections are described in the following section.

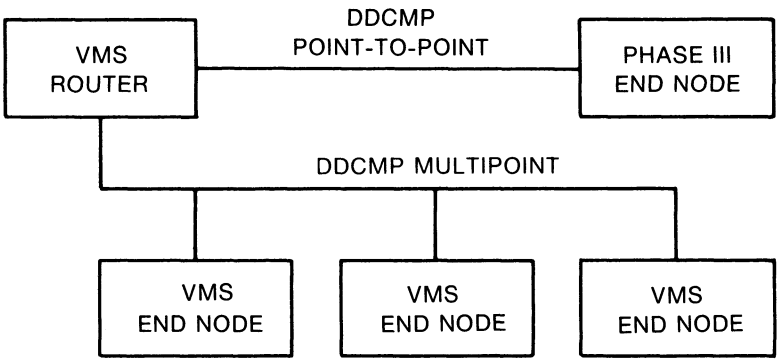
1.3.2.1 DDCMP Point-to-Point and Multipoint Connections

A **point-to-point** configuration consists of two systems connected by a single communication channel. Figure 1–3 illustrates DDCMP point-to-point and multipoint configurations.

A **multipoint** configuration consists of two or more systems connected by a communications channel, with one of the systems (called the **control station**) controlling the channel. All other systems on the communications channel are known as **tributaries**. (Note that, if only two systems are connected in a multipoint configuration, one is the master and one is the tributary. However, this is not a very efficient use of the communication channel.) The control station is responsible for telling the tributaries, in turn, when they may use the channel; this procedure is known as **polling**. Tributaries are not allowed to use the channel until they are polled. The control station, however, may use the channel whenever it is available. Also, the tributaries on a multipoint line are not allowed to communicate directly with each other, but only through the master.

Point-to-point circuits and multipoint circuits perform as virtual circuits: nodes on these circuits interact as though a specific circuit were dedicated to them throughout the transmission; in fact, however, the actual physical connection is allocated by the routing mechanism. Initialization of nodes on DDCMP circuits involves guaranteed delivery of routing messages. Also, individual nodes on DDCMP circuits must be addressed directly; no multicast or broadcast addressing capability is available as with an Ethernet circuit.

Figure 1–3 Typical DDCMP Point-to-Point and Multipoint Connections



1.3.2.2 Synchronous DDCMP Connections

You use synchronous communications devices for high-speed point-to-point or multipoint communication (for example, connecting two VAX–11/780 systems).

The synchronous DDCMP protocol can run in full- or half-duplex operation. This allows DDCMP the flexibility of being used for local synchronous communications, or for remote synchronous communications over a telephone line using a modem. DDCMP has been implemented in microcode in such devices as the DMC11 and DMR11 to run at speeds up to one megabit per second in a point-to-point configuration. The DDCMP multipoint protocol (point-to-point also) has been implemented in microcode in the DMP11 device to run at speeds up to 500 kilobits per second. For the DMF32, DDCMP has been implemented in the driver software for the synchronous communications port.

1.3.2.3 Asynchronous DDCMP Connections

Asynchronous connections provide for low-speed, low-cost, point-to-point communication (for example, as an inexpensive way of connecting a MicroVAX system to a VAX–8000 series system). Asynchronous DDCMP is implemented in software and can be run over any directly connected terminal line that the VMS system supports. The asynchronous DDCMP protocol provides for a full-duplex connection and can be used for remote asynchronous communications over a telephone line using a modem. Asynchronous connections are not supported for maintenance operations or for controller loopback testing.

You can make two kinds of asynchronous connections over the network:

- A static connection: the asynchronous line is permanently configured as a communications device
- A dynamic connection: a line connected to a terminal port is switched to an asynchronous communications line for the duration of a call

1.3.2.4 Static Asynchronous Connections

A static asynchronous DDCMP connection is a permanent DECnet connection between two nodes physically connected by terminal lines. You convert the terminal lines to static asynchronous DDCMP lines by issuing commands to set the lines to support the DDCMP protocol. The user at each node then turns the appropriate circuits and lines on for DECnet use. After the communications link is established, it remains available until a user turns off the circuit and line and clears the entries from the DECnet database.

Static asynchronous DDCMP configurations require the asynchronous DDCMP driver to be connected. The asynchronous DDCMP protocol can run in full-duplex operation on local asynchronous communication devices. Examples of these devices are the DZ11 and the DMF32 asynchronous communications port.

You can configure a dialup line as either a static or dynamic asynchronous line, but may find the dynamic connection more secure and convenient to use.

1.3.2.5 Dynamic Asynchronous Connections

A dynamic asynchronous connection is a temporary connection between two nodes, generally over a telephone line using modems. The terminal lines at both ends of the connection can be switched to asynchronous DDCMP communications lines and then switched back to terminal lines.

You can use dynamic asynchronous connections to establish a DECnet link to another computer for a limited time or to create links to different computers at different times.

For example, as a user of a personal computer (non-VMS), you can cause a dynamic asynchronous connection to be made for the length of the telephone call to a VAX–8000 series system. You must first establish a process on your system as a **terminal emulator** (enabling the remote connection to look like a local connection). You dial in over a telephone line to a process on the other system (which is established as a **virtual terminal**) and log in. You can then enter a command that causes the terminal lines at each end of the connection to be switched to DDCMP mode for DECnet use. When you hang up the telephone or turn off the circuit, the lines are automatically switched back to terminal lines.

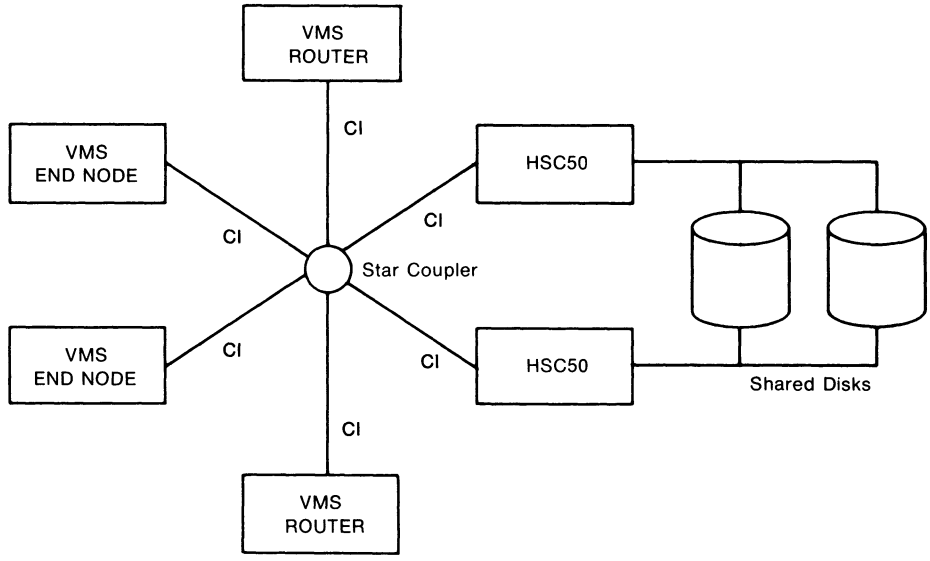
Security measures provide protection against a caller at an unauthorized node forming a dynamic asynchronous connection with another node (see Section 2.10.6). Before a dialup node can establish a dynamic connection with a remote node, the remote node verifies that the dialup node is authorized to make a connection. It checks that the node is of the appropriate type (router or end node), and, without revealing its own password, verifies the routing initialization password sent by the dialup node. Also, for increased security, the connection is ended automatically when the telephone is hung up.

You can establish a dynamic asynchronous connection over a hardwired terminal line. The connection is maintained for the duration of the DECnet session. The dynamic connection permits the system to be used as a terminal emulator when not switched to DECnet use.

1.3.3 DECnet–VAX Configurations for VAXclusters

A VAXcluster is an organization of VMS operating systems that communicate over a high-speed communications path, the CI, and share processor resources as well as disk storage. Figure 1–4 shows a typical VAXcluster. The CI is the physical link between the nodes in a VAXcluster. The CI cables from the individual nodes in the cluster are connected to a star coupler. The HSCs are hierarchical storage controllers that enable VAXcluster nodes to share disks.

Figure 1–4 Typical VAXcluster Configuration with CI as a Data Link



DECnet–VAX connections are required for all VMS operating systems in the VAXcluster. Use of DECnet–VAX ensures that VAXcluster system managers can access each node in the cluster from a single terminal, even if terminal-switching facilities are not available. DECnet–VAX is also required by the User Environmental Test Program (UETP).

The choices for DECnet–VAX physical links for use in the VAXcluster are as follows:

- Connecting each VMS node in the cluster to an Ethernet (as shown in Figure 1–2)
- Using the CI that connects the cluster nodes as the DECnet–VAX data link (as shown in Figure 1–4)

Connecting each VAXcluster node to an Ethernet provides distinct advantages:

- Each node in the cluster can be an end node, resulting in lower overhead for these nodes, decreased routing traffic throughout the network, and simpler installation procedures. Note that there must be at least one router on the Ethernet to which the cluster end nodes are attached.

- Ethernet provides for better performance in DECnet transmissions than the CI, despite the higher data link **bandwidth** of the CI, because the Ethernet communications protocol allows larger buffer sizes.
- Terminal servers can be used when nodes in a VAXcluster are connected to an Ethernet. DIGITAL's terminal servers offer a number of benefits to the VAXcluster user, such as load balancing and easier cluster management.

If you use only one physical link to connect each cluster node to the network, you should use the Ethernet link instead of the CI data link, because of the better DECnet performance of the Ethernet. In this case, the CI should perform the functions of a system bus and not be enabled as a DECnet data link.

A VAXcluster node connected to an Ethernet may require additional DECnet links in order to communicate with remote nodes not on the Ethernet. You must configure a VAXcluster node connected to more than one DECnet link as a router, not as an end node.

If the nodes in the VAXcluster are not connected to an Ethernet, the CI should be used as the DECnet data link between the nodes. CI circuit devices are configured as though they were multipoint devices, but each node on the CI can talk directly to every other node and no polling is involved.

A two-node VAXcluster that uses the CI as the data link can be configured using end nodes. If additional nodes are configured in the cluster, however, at least one router is required. The CI does not have a broadcast capability (such as that of the Ethernet). Thus, the router is needed so that the nodes in the cluster can identify each other. If the router in a three-node cluster fails, the cluster reverts to being a two-node cluster and can consist of end nodes only. You can use network management commands to create a circuit between the end nodes. For a cluster of four or more nodes, more than one router is required in order to prevent the loss of communications capability between the remaining nodes if one router fails. Also, backup circuits can be provided between end nodes in case of router failure.

A VAXcluster can be configured so that the whole cluster appears to other network nodes as though it were a single node, with an address different from that of any DECnet node within the VAXcluster. This address usually has a node name associated with it. Thus, you can access the VAXcluster as a whole by an **alias node identifier**, which can be either its node name or its node address. All or some of the nodes in a VAXcluster can elect to use this special node identifier as an alias, while retaining their unique individual node names and addresses. Each node that assumes the alias node identifier can specify whether it will accept incoming connections directed to the alias address. It can also specify the network services for which the cluster alias node identifier is to be used on outgoing connections and the network services that will accept incoming calls.

At least one of the nodes in the cluster that accepts the alias node identifier must be a router. The router informs other nodes in the network of the alias node address for the cluster. When the router receives packets addressed to the alias node address, it forwards them to the appropriate nodes in the cluster. The cluster alias node identifier can be very useful in network operations involving shareable resources. Network users outside the VAXcluster can access cluster resources without knowing which nodes are active in the cluster. For example, if a user on a cluster node sends a MAIL message, it does not matter whether that particular node is active when a reply to the message is received.

1.3.4 X.25 Network Configurations

Packet switching data networks provide fast, dependable communications between geographically distributed nodes. Data transmitted over a PSDN is divided into packets, each of which has a header containing control and destination information. The PSDN interleaves packets from many users over shared transmission lines and delivers the packets in the correct order to the proper destinations. The routing of packets through the PSDN is handled by the PSDN itself and is invisible to the user.

In X.25 network terminology, your computer or terminal is called **data terminal equipment (DTE)** and the PSDN interface to which it is connected is known as **data circuit-terminating equipment (DCE)**. The DTE can operate in packet mode or in character mode. A character-mode terminal is also known as an X.29 terminal.

1.3.4.1 X.25 and X.29 Recommendations

Recommendations for standard network interfaces for DTEs have been established by the CCITT (Comite Consultatif International Telegraphique et Telephonique). The X.25 recommendation defines the interface between the packet-mode DTE and the DCE. The X.25 recommendation defines three levels of protocols for this interface: Level 1 covers physical and electrical characteristics; Level 2, link access procedures; and Level 3, packet procedures.

The X.29 recommendation defines the procedures for information exchange between a packet-mode DTE (a computer) and the packet assembly/disassembly (PAD) facility of the PSDN or host.

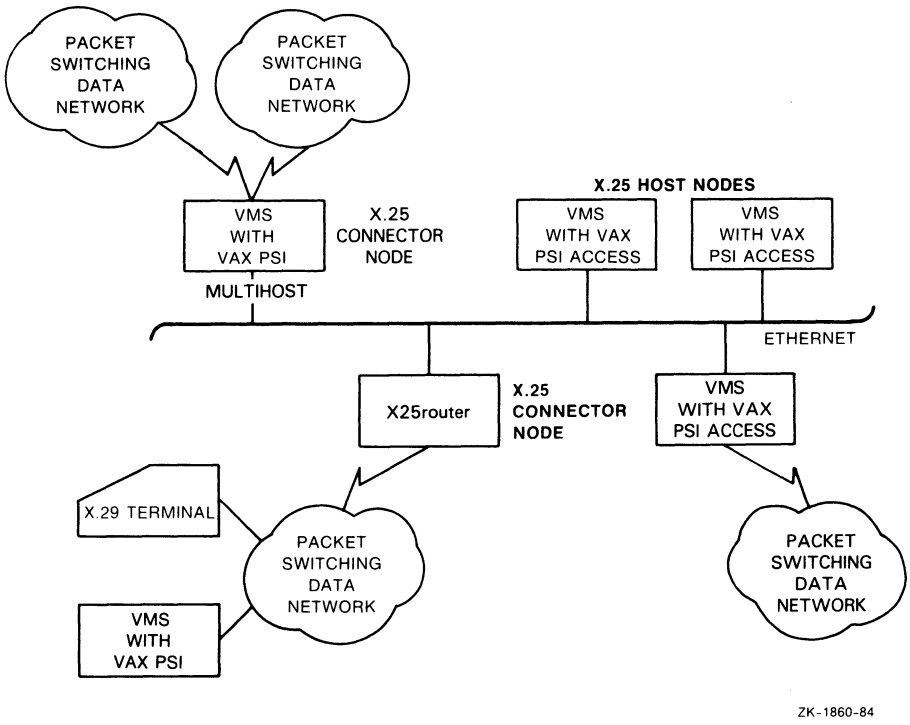
Communication between a local DTE and a remote DTE is by means of a X.25 virtual circuit, a logical association between the two DTEs set up specifically to handle the exchange of data between them. An X.25 virtual circuit can be permanent or temporary. The **permanent virtual circuit (PVC)** is similar to a leased line. The temporary or **switched virtual circuit (SVC)** is similar to a dialup line and requires calls to be set up and cleared. A local DTE is connected to the PSDN by a synchronous X.25 line, over which X.25 virtual circuits operate. Examples of X.25 line interfaces are the DUP11 and the DMF32 synchronous line unit.

As well as supporting various PSDNs, VAX PSI can also support an ISO standard 8208 network. ISO 8208 is the International Standards Organization's definition of the CCITT X.25 recommendations.

1.3.4.2 X.25 Connections

Figure 1–5 illustrates typical X.25 connections that permit a VMS node to communicate with a remote node over a PSDN.

Figure 1–5 X.25 Connections in a DECnet Network Configuration



You can configure a VMS node to communicate over PSDNs in the following ways:

- By means of a data link mapping (DLM) circuit, which is an X.25 virtual circuit used as a DECnet data link (provided the remote node runs PSI and DECnet–VAX or DECnet–RSX, or is an X25router)
- Through a direct connection to a PSDN over an X.25 virtual circuit (this is called native-mode operation)
- As a host node using an X.25 connector node to access a PSDN. A connector node may be a VMS node with VAX PSI configured in multihost mode or it may be an Ethernet communications server, such as the X25router.

You must have VAX PSI software installed on your DECnet–VAX node to configure your node for VAX PSI native-mode operation or multihost mode operation. Native-mode operation permits incoming and outgoing calls to be made between the local node and a remote DTE. Multihost mode operation allows the local node to be a connector node or gateway, which host nodes can use to communicate over a PSDN with remote DTEs. Each host node must have VAX PSI Access software installed.

Each native mode node and each connector node may support several DTEs. The DTEs in turn may be connected to different PSDNs and provide multiple connections to each PSDN. Each host may connect to several different connector nodes.

1.4 Managing the Network

As system manager of a VMS operating system, you can use a network management utility program to configure the system as a DECnet–VAX node in the network, and perform network management and maintenance functions for your own node and other nodes in the network. The following subsections summarize network management functions.

1.4.1 Network Control Program

To configure, control, monitor, and test the network, you use the Network Control Program (NCP), a VMS utility program. The following types of users employ NCP:

- Users of both DECnet–VAX and VAX PSI. These users can employ all NCP commands.
- Users of DECnet–VAX only. DECnet–VAX users employ all NCP commands except those that relate to X.25 **modules**.
- Users of VAX PSI only. PSI native-mode and multihost-mode users (who are not using DECnet–VAX circuits) employ only NCP commands that relate to X.25 modules, circuits, lines, objects, and **logging**.

The network **components** the system manager configures are listed in Section 1.4.5 and described in detail in Chapter 2. Chapter 3 discusses how to use NCP commands and parameters to perform network management. The NCP commands and parameters and guidelines for using them, including restrictions on the use of individual NCP parameters, are specified in the *VMS Network Control Program Manual*.

1.4.2 Network Management Responsibilities

As system manager of a DECnet–VAX network, you have a number of key responsibilities, which include the following:

- Defining network components and their parameters in a central **configuration database** at the local node and, optionally, at remote nodes. (The **local node** is the node at which you are physically located; a **remote node** is any node other than the local node in your network.)
- Coordinating with the system managers of other nodes in the network to ensure uniform assumptions about network parameter settings such as circuit cost.
- Configuring your node to ensure proper network routing operation and updating VMS SYSGEN procedures to allow enough space for the networking software.
- Controlling and monitoring local and remote network operation.
- Testing network hardware and software operation.
- Loading systems downline to unattended remote nodes.
- Connecting to an unattended remote node to serve as its console.

If your network includes VAX PSI, you have the following additional responsibilities:

- Defining VAX PSI components and their parameters in the network configuration database and thus configuring VAX PSI.
- Monitoring the operation of VAX PSI using PSI management utilities.
- Analyzing hardware and software operation and diagnosing problems related to PSI operation.

The following sections outline the network-related tasks that you perform as system manager and describes several of the facilities DECnet–VAX and VAX PSI provide to perform those tasks.

1.4.3 DECnet–VAX Licenses and Keys

To enable your node to communicate with other nodes in the DECnet network, you need a DECnet–VAX license and key. You must purchase either a full function or an end node license, and enable the license by registering the appropriate DECnet–VAX key on your system. You register DECnet–VAX keys by using the License Management Facility (LMF). To register the key, you use the License Management Utility (LICENSE) to enter the information from the LMF Product Authorization Key (PAK).

The DECnet–VAX full function key allows the node on which it is enabled to be configured as either a routing node or an end node. The end node key permits the use of the DECnet–VAX end node capability only.

If you have purchased a DECnet–VAX end node license and now require the additional DECnet–VAX full function capability, you may purchase a DECnet–VAX end node to full function upgrade license and key.

Note that you do not need a DECnet–VAX license or key if you are planning to only use VAX PSI in native mode (as long as you do not want to use data link mapping to communicate with other DECnet nodes).

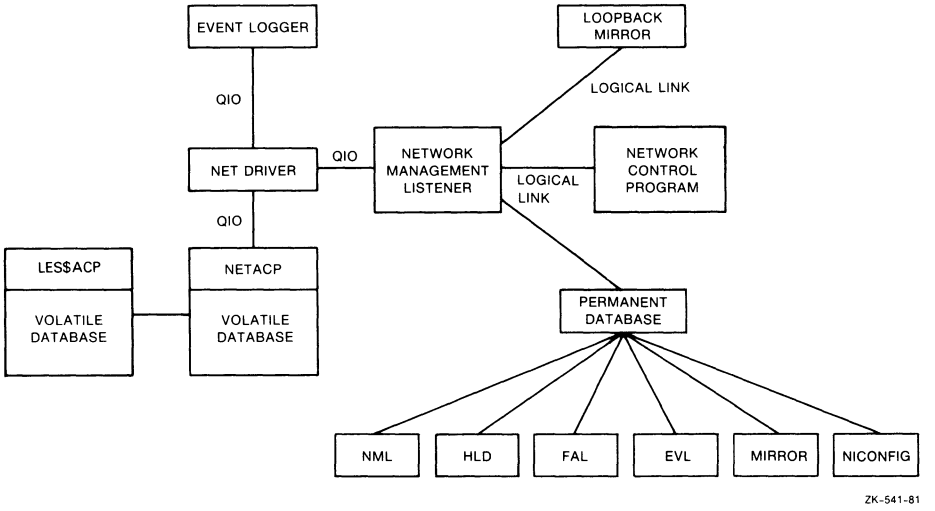
1.4.4 DECnet–VAX and VAX PSI Network Management Software

Figure 1–6 displays the DECnet–VAX and VAX PSI software that the system manager uses to configure, control, and monitor the network.

Network management software components are as follows:

- Ethernet configurator (NICONFIG), a network image that listens to system identification messages on Ethernet circuits, and maintains a user-accessible database of configuration information on all systems on the Ethernet.
- Event logger (EVL), an image that logs significant **events** to provide information to the system manager for possible intervention or future reference.
- File access listener (FAL), a network image that receives and processes remote file access requests for files at its node on behalf of remote users.

Figure 1–6 DECnet–VAX and VAX PSI Software



- Host loader (HLD), an image that communicates with the DECnet–RSX Satellite Loader (SLD) to load tasks downline to an RSX–11S node.
- Loopback mirror (MIRROR), a network image that participates in Network Service Protocol (NSP) and Routing layer loopback testing.
- Network ancillary control process (NETACP), a VMS ancillary control process that controls all lines and circuits, maintains a picture of the network topology, and creates a process to receive inbound logical link connection requests.
- Network Control Program (NCP), an interactive utility program that permits you to control and monitor the network.
- Network driver (NETDRIVER), a VMS pseudo-device driver that provides logical link and routing services. It implements NSP and Routing, and provides a user process with a Queue I/O (QIO) interface to a logical link service.
- Network management listener (NML), an image that receives network management commands, such as NCP commands, from the Network Management layer through the Network Information and Control Exchange (NICE) protocol. NML performs all local network management functions as well as control and information functions requested by remote nodes. NML spawns a subprocess, the maintenance operation module (MOM), for maintenance functions such as downline load, upline dump, and loopback testing.
- LES ancillary control process (LES$ACP), a process that supports several LES-based communications products. It is used by VAX PSI to control all X.25-related functions. LES$ACP has the VAX PSI volatile database.
- Permanent database, a collection of disk-resident files that define the network as known to the local node. If VAX PSI is configured in the network, a subset of the permanent database is maintained as the VAX PSI permanent database for the local DTEs.

- Volatile database, maintained by NETACP, a memory-resident database containing current network configuration parameters. If VAX PSI is configured in the network, a subset of the volatile database is maintained as the PSI volatile database in PSIACP for the local DTEs.

Many of these software components are user-transparent processes over which the system manager has no control. This manual describes DECnet–VAX and VAX PSI software only as it serves to highlight and clarify the functions and operation of NCP. The various DNA specifications describe the different protocols that facilitate network communication.

1.4.5 Configuring a Network

The system manager must configure each DECnet–VAX node and VAX PSI DTE as part of the network.

1.4.5.1 Configuring a DECnet–VAX Node

At the outset, the system manager is responsible for configuring the network from the perspective of local node network operation. This involves supplying information at the local node about various network components such as nodes, circuits, lines, and objects. This information constitutes what is called the configuration database for the local node. Each node in the network has such a database. You supply information about the configuration database through NCP.

If you are configuring a DECnet–VAX node for the first time or want to rebuild the configuration database for your local node, you can use the interactive NETCONFIG.COM procedure to configure your node automatically. To update an existing node database to contain current information about other nodes in the network, you can copy the information from the node database of another node to which you have access.

Chapter 3 discusses the function of the configuration database and the general use of NCP and most NCP commands. Chapter 5 describes how to use the NETCONFIG.COM procedure to configure your node automatically, and presents sample configuration commands for various network configurations. The *VMS Network Control Program Manual* contains a summary description of NCP operation, command prompting, and the syntax of all NCP commands.

1.4.5.2 Configuring VAX PSI DTEs

If VAX PSI is to be run, the system manager is responsible for installing and configuring VAX PSI for the local DTEs. Configuring VAX PSI involves supplying information about various VAX PSI components, such as circuits, lines, modules, and objects. The information is contained in the PSI configuration database for the local node and, if both DECnet–VAX and VAX PSI are configured, in the DECnet–VAX configuration database for the local node. You use NCP commands to supply information to the configuration database.

If your node is to serve as an X.25 multihost connector node to provide access to PSDNs for host nodes, you must configure VAX PSI software in multihost mode. If your node is to be a host node that uses the connector node to access a PSDN, you need to install and configure only the VAX PSI Access software. The procedures for configuring VAX PSI software or VAX PSI Access software on your DECnet–VAX node are described in Chapter 5.

1.4.5.3 A Network Topology

Figure 1–7 illustrates a hypothetical network topology made up of various DIGITAL operating systems. Figure 1–8 illustrates the same topology for a network that has been divided into multiple areas. These examples are referred to as the "network examples" throughout this manual.

Figure 1–7 Topology of a Single-Area DECnet Network

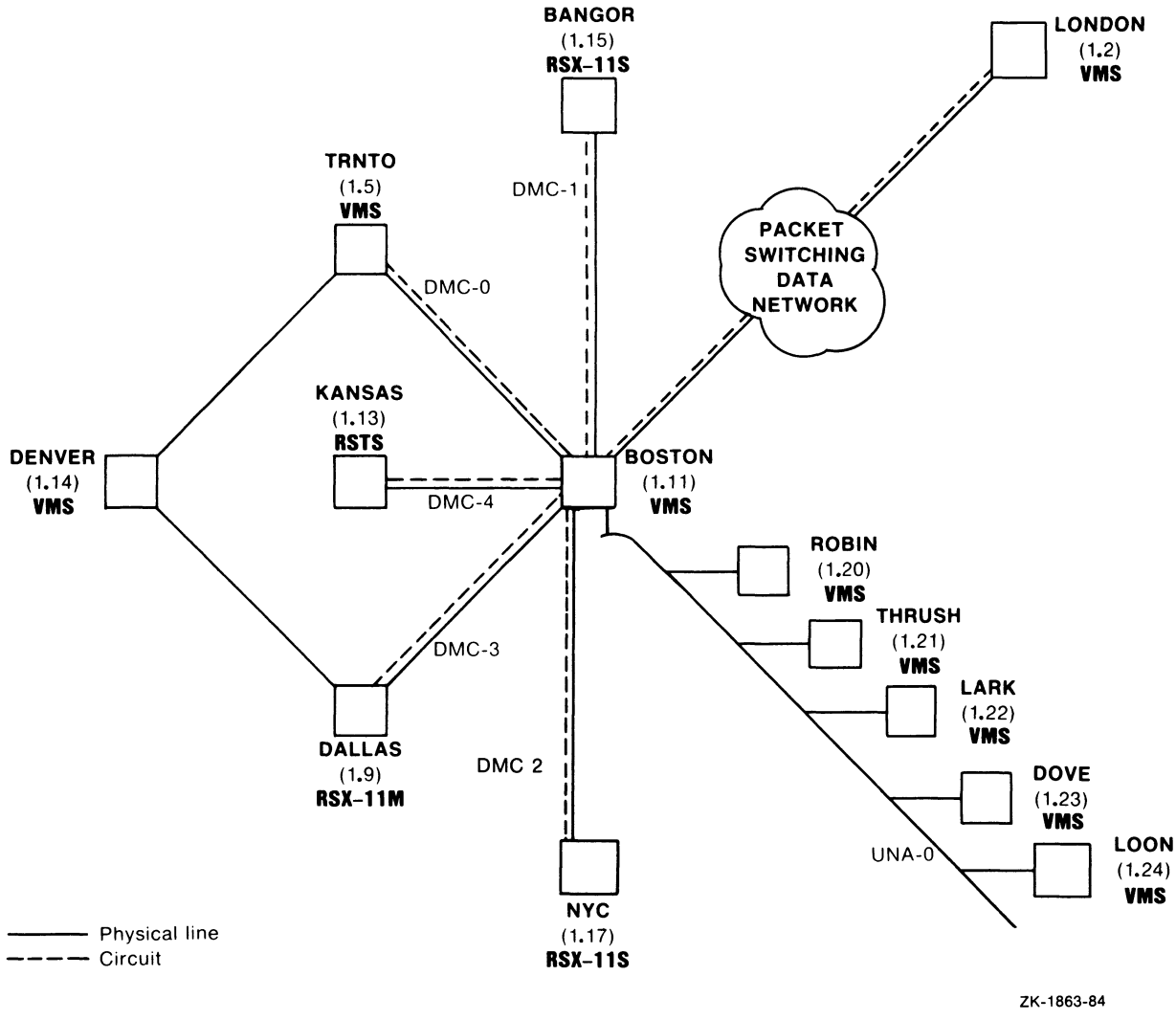
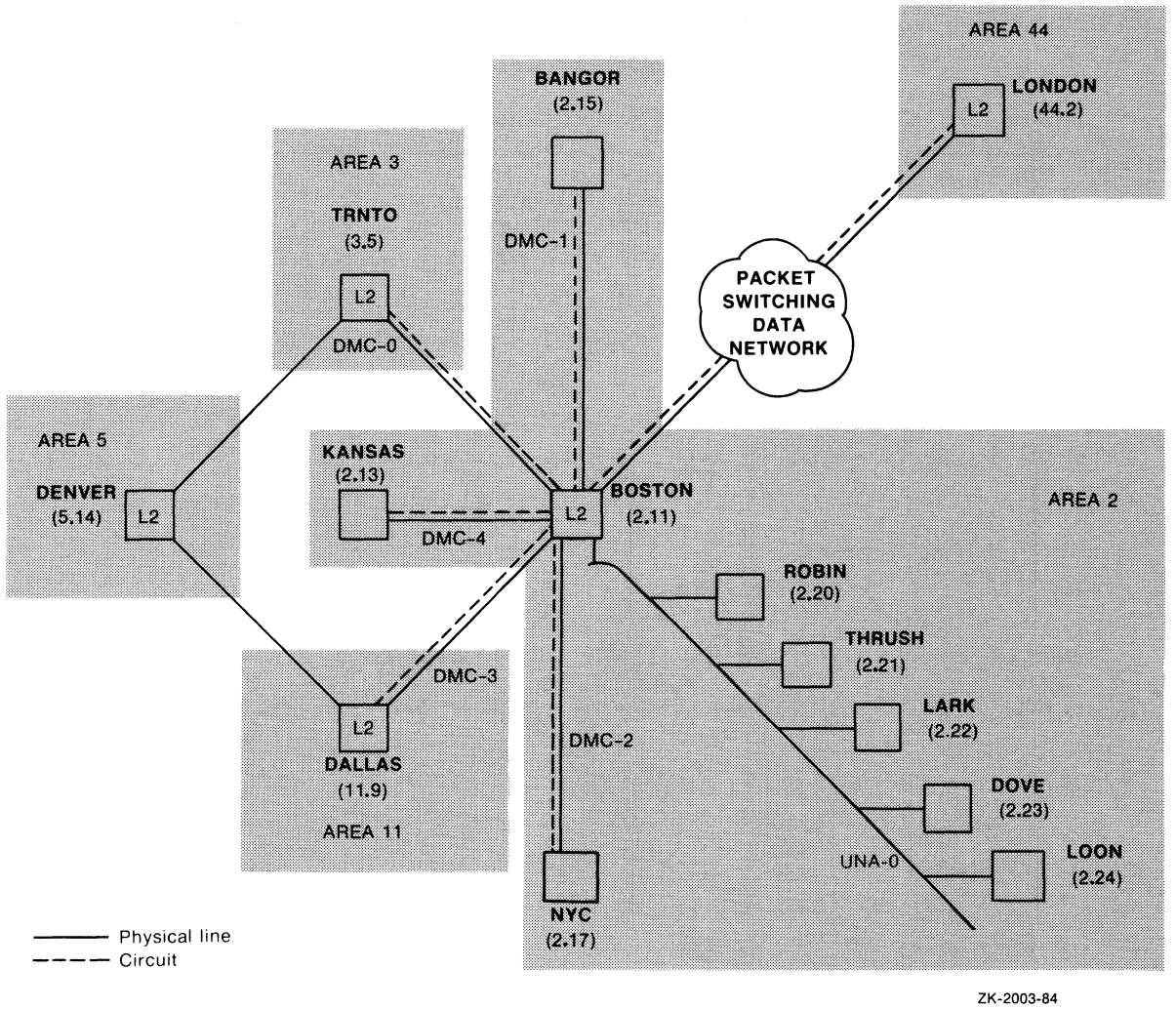


Figure 1–8 Topology of a Multiple-Area DECnet Network



Figures 1–7 and 1–8 show some, but not all, of the network components about which the system manager gathers and consolidates information in the configuration database. Using NCP, you can control the following six network components:

- Nodes. Nodes are DIGITAL operating systems using DECnet software to communicate with other operating systems across the network.
- Modules. VAX PSI modules include the X.25 protocol module, which performs data packet handling and multiplexing of X.25 circuits over lines to PDSNs; and the X.25 and X.29 server modules, which handle X.25 and X.29 calls, respectively. DECnet–VAX modules include the X.25 access module, which permits a VMS host node to communicate over a PSDN by means of a connector node, and the Ethernet configurator module, which lists all nodes on the Ethernet.

- Circuits. Circuits are virtual communications paths between nodes and between DTEs. Circuits operate over physical lines and are the medium on which all I/O occurs. DECnet processes "talk" over circuits by means of logical links. These links carry a single stream of full-duplex traffic between two user-level processes. There can be multiple logical links on each DECnet circuit.
- Lines. Lines are physical data paths between nodes, or between DTEs and DCEs (X.25 network interfaces).
- Objects. Objects are processes that receive logical link requests. They perform specific network functions. An example is FAL, which is used for remote file access. Objects also receive incoming X.25 calls.
- Logging. Logging is a network feature that enables the automatic recording of useful network events that occur during network operation.

These components, the DECnet and PSI software modules and databases, and the hardware make up the network. NCP command examples in this manual relate to the components illustrated in the network example.

1.5 User Interface to the Network

This section describes the user interface to the DECnet–VAX network. It includes a general description of operations that you can perform over the network and a list of the programming languages that you can use for designing network applications. The following sections present general information that you need to know to access the DECnet–VAX network.

1.5.1 Performing Network Operations

You can use the DECnet–VAX software to perform a variety of operations over the network:

- Manipulate files on remote nodes (for example, transfer, delete, or rename files).
- Access remote files at the record level.
- Perform task-to-task communications.

DECnet–VAX allows you to access files on remote nodes as though they were on your local node. It also allows you to design applications that communicate with each other over the network. For detailed information about remote file access and task-to-task communication, including examples of each type of network application, see Chapter 8.

Throughout this document, the term **task** refers to an image running in the context of a process, the term **local** refers to the node at which you are located physically, and the term **remote** refers to the node with which you establish a connection. Note that, in certain situations such as testing, you can establish a logical link between two processes on the same node.

The VMS operating system and DECnet–VAX communications software are integrated to provide a high degree of transparency for user operations. For some applications, however, it is desirable (and sometimes necessary) to have more direct access to network-specific information and operations. For this purpose, DECnet–VAX provides nontransparent communication.

The following sections describe some of the general transparent and nontransparent features of DECnet–VAX in terms of the user interface to the network. For more detailed information, including examples of transparent and nontransparent DECnet–VAX applications, see Chapter 8.

In addition to remote file access and task-to-task communication, DECnet–VAX also allows you to communicate with remote nodes through the heterogeneous command terminal facility (SET HOST), described in Chapter 8.

When designing user applications to perform network operations, you can use standard DCL commands, higher-level language I/O statements, VMS RMS service calls, and system service calls.

1.5.1.1 Designing User Applications for Network Operations

DECnet–VAX supports several programming languages for network applications:

- DCL commands and command procedures
- Higher-level language programs
- MACRO programs using RMS service calls or system service calls

You can use several higher-level languages to develop networking applications, including VAX Ada, VAX FORTRAN, VAX BASIC, VAX BLISS, VAX PASCAL, VAX C, VAX PL/I, and VAX COBOL. With any of these languages, you can access remote files and create tasks that exchange information across the network.

Table 1–1 summarizes the normal use of the programming languages for specific network operations that you can perform with DECnet–VAX.

Table 1–1 Network Access Levels

User Language	Network Operation	Language Calls	Access Level
DCL	Network command terminals	DCL commands	Transparent network access using DCL
	Remote file manipulation		
	Task-to-task communication		
Higher-level languages	Remote file access (files and records)	Higher-level language I/O statements	
	Task-to-task communication		Transparent network access using RMS
MACRO or higher-level languages	Remote file access (files and records)	RMS service calls	
	Task-to-task communication		
MACRO or higher-level languages	Task-to-task communication	System service calls	Transparent and nontransparent network access using QIO

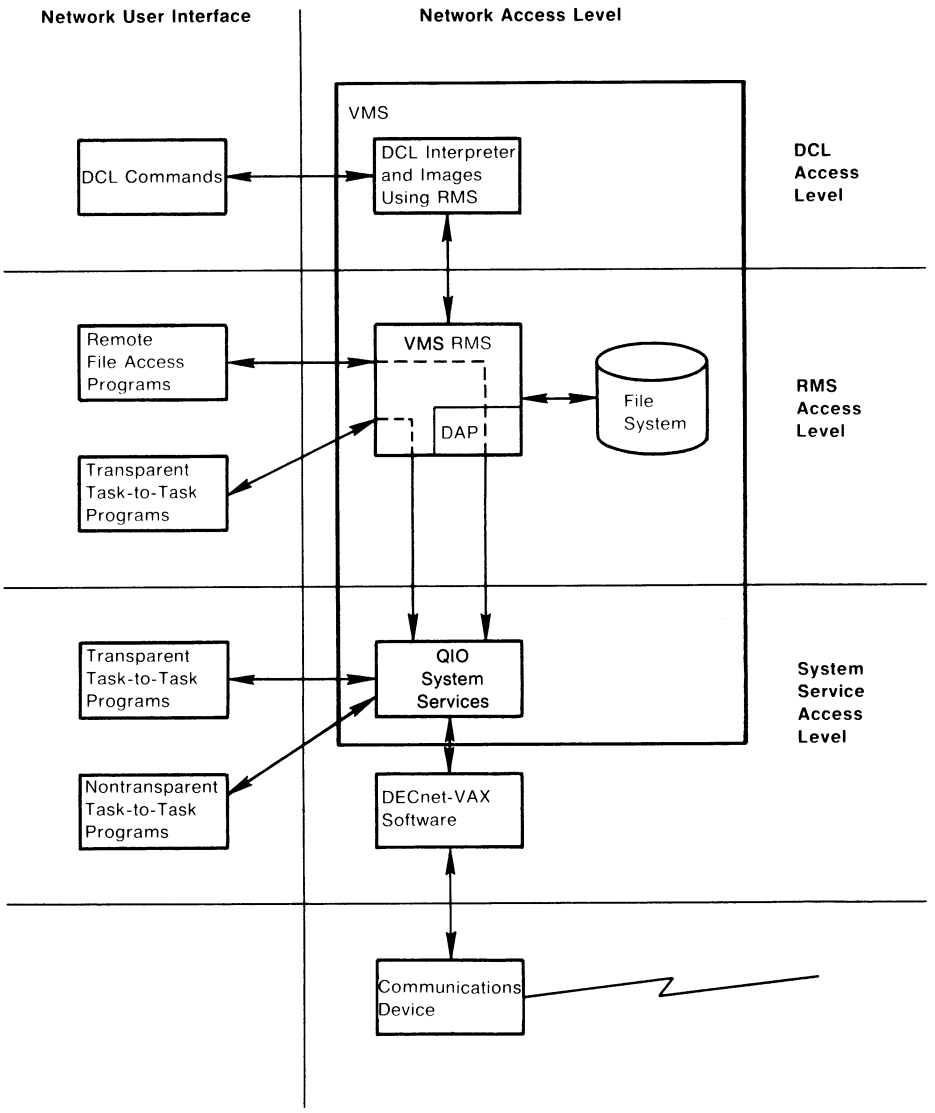
1.5.1.2 Choosing a Language for a Specific Network Application

The way you access the network is directly related to the language you use and the network operation you perform. For example, you may want to use standard VMS RMS calls in a VAX MACRO program to access remote files, then use system service calls to communicate between MACRO programs in a task-to-task communication application. Figure 1–9 shows three access levels and the corresponding network operations. The various levels of network access provide a convenient context in which to discuss typical user operations over the network.

The first two levels of access, DCL and RMS, are entirely transparent to the network user. Because you use standard DCL commands and RMS service calls to access remote files, no DECnet-specific calls are required at these levels of access. You need only specify in your file specification the remote node on which the file resides. Likewise, higher-level language tasks can use a variation of the standard VMS file specification in conjunction with standard I/O statements to access remote tasks and exchange information; thus, this form of task-to-task communication is also transparent. As with device-independent I/O operations, transparent network access allows you to move data across the network with little concern for the way this operation is performed.

The third level of access, system services, provides both a transparent and a nontransparent user interface to the network. Transparent communication at the system-service level provides all the basic functions necessary for two tasks to exchange messages over the network. As with the higher-level language I/O interface, these operations are transparent because they do not require DECnet-specific calls. Rather, you use standard system service calls to implement them. Nontransparent communication extends this basic set of functions to allow a nontransparent task to receive multiple inbound connections and to use additional network protocol features such as optional user data and interrupt messages. As with device-dependent I/O, nontransparent communication allows you to exploit certain network-specific characteristics to coordinate a more controlled communication environment for exchanging information.

Figure 1–9 Network Access Levels and DECnet–VAX User Interface



1.5.2 Accessing the Network

This section presents general information that you need to know to access the network by means of DECnet–VAX software. This information covers network file and task specifications, access control parameters, and how to use logical names in network applications.

The format for file specifications is applicable to file-handling operations for both the DCL and the RMS interfaces to the network. The task specification format pertains to task-to-task communication. The information on access control is significant because it defines the way that both local and remote nodes grant access to their system resources.

1.5.2.1 Using File and Task Specifications in Network Applications

DECnet–VAX uses the standard VMS file specification format for remote file-handling applications. A node specification string that includes a node name must be present. You can also include an optional **access control** string in the node specification to specify explicitly the user name and password of a specific **account** to use on the remote system. For example:

```
TRNTO"SMITH JOHN"::WORK_DISK:TEST.DAT;1
```

This file specification contains explicit access control information and can be used to access the file TEST.DAT, which resides in user Smith's top-level directory on the device WORK_DISK on node TRNTO.

The following file specification, which does not contain explicit access control information, can also be used to access the remote file TEST.DAT, provided a proxy or default nonprivileged DECnet account exists on the target node:

```
TRNTO::DBA1:[SMITH]TEST.DAT;1
```

For more information about file specification strings, including format examples, see the *VMS DCL Concepts Manual*.

Task-to-task communication requires the use of a **task specification string** enclosed in quotation marks. This string identifies the target task to which you want to connect on a remote node. For example:

```
BOSTON::"TASK=TEST2"
```

This task specification string identifies the task TEST2 by means of the TASK= form of task specification. You can also use the 0= form to specify a task. For example:

```
BOSTON"JONES KC"::"0=TEST2"
```

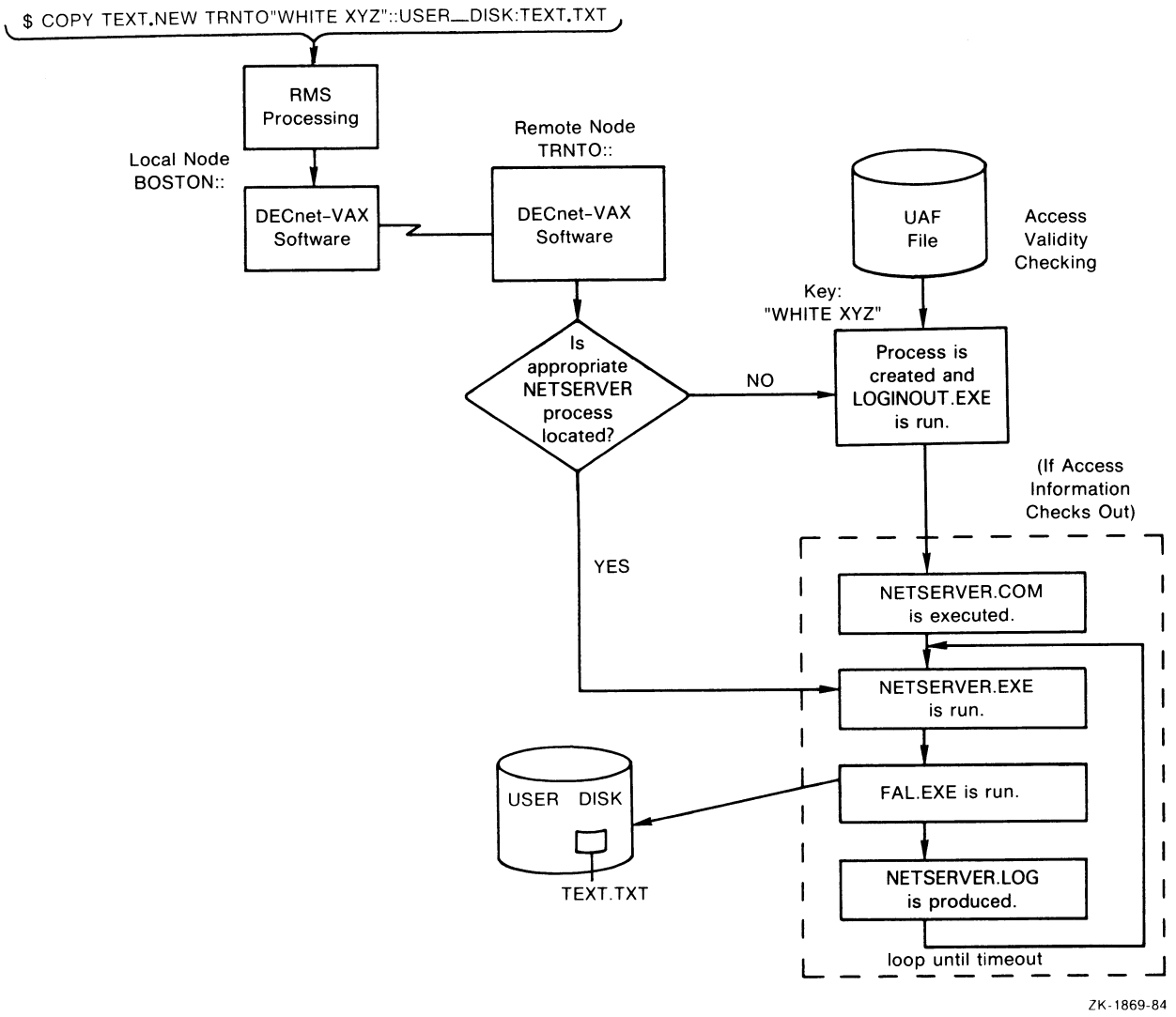
This task specification string also identifies the task TEST2. Note that, in this case, explicit access control information is also included in the node specification string. For more information about task specifications, see Chapter 8.

1.5.2.2 Using Access Control for Network Applications

Access control is the control that a node exercises over inbound logical link connections. The terms **inbound** and **outbound** refer to the direction of the logical link connection request. A node receives and processes inbound requests; it processes and sends outbound requests. This distinction is useful for discussing access control as it relates to VMS nodes in a network. If the node to which you want to connect is not on a VMS operating system, refer to appropriate DECnet documentation.

When DECnet–VAX software sends an outbound connection request in response to either a remote file access or a task-to-task communication operation, you may need certain access control information to connect successfully to the remote node and to log in. As in logging in at your local VMS node, you can supply specific access control information in the form of a user name and password that the remote node recognizes. The remote node processes inbound connection requests containing this information to verify that you are a valid user of the system. For more information about inbound and outbound connection requests, see Section 2.10.2. Figure 1–10 illustrates the access control processing that takes place for a DCL command.

Figure 1–10 Remote File Access Using Access Control String Information



When you do not provide explicit access control information in the connection request, DECnet–VAX software uses the remote node name specified in the connection request as a key to locate the appropriate record in the local configuration database. This record contains default access control information applicable to the remote node. Your system manager creates this entry when establishing the configuration database. (For additional information about the configuration database, refer to Chapter 3.)

Depending on the privileges required by the object to which you want to connect and those of the user process (see Figure 1–11), one of three possible sets of default access control information is sent to the remote node: default **privileged**, default **nonprivileged**, or null. Because these defaults are node parameters, all privileged operations requested with default access control for a given node run under the same default account. The same is true for nonprivileged operations requested with default access control.

If the target node is running DECnet–VAX, it can associate incoming connect requests with specific accounts other than the default nonprivileged DECnet account. This type of access is known as **proxy login** and requires the originator of the request to have a proxy account on the target node and proxy login access to be enabled at that node. Proxy login is described in Section 2.10.5. Figure 1–11 illustrates the access control processing that takes place for the same DCL command as in the example in Figure 1–10, except that the DCL command does not specify an access control string.

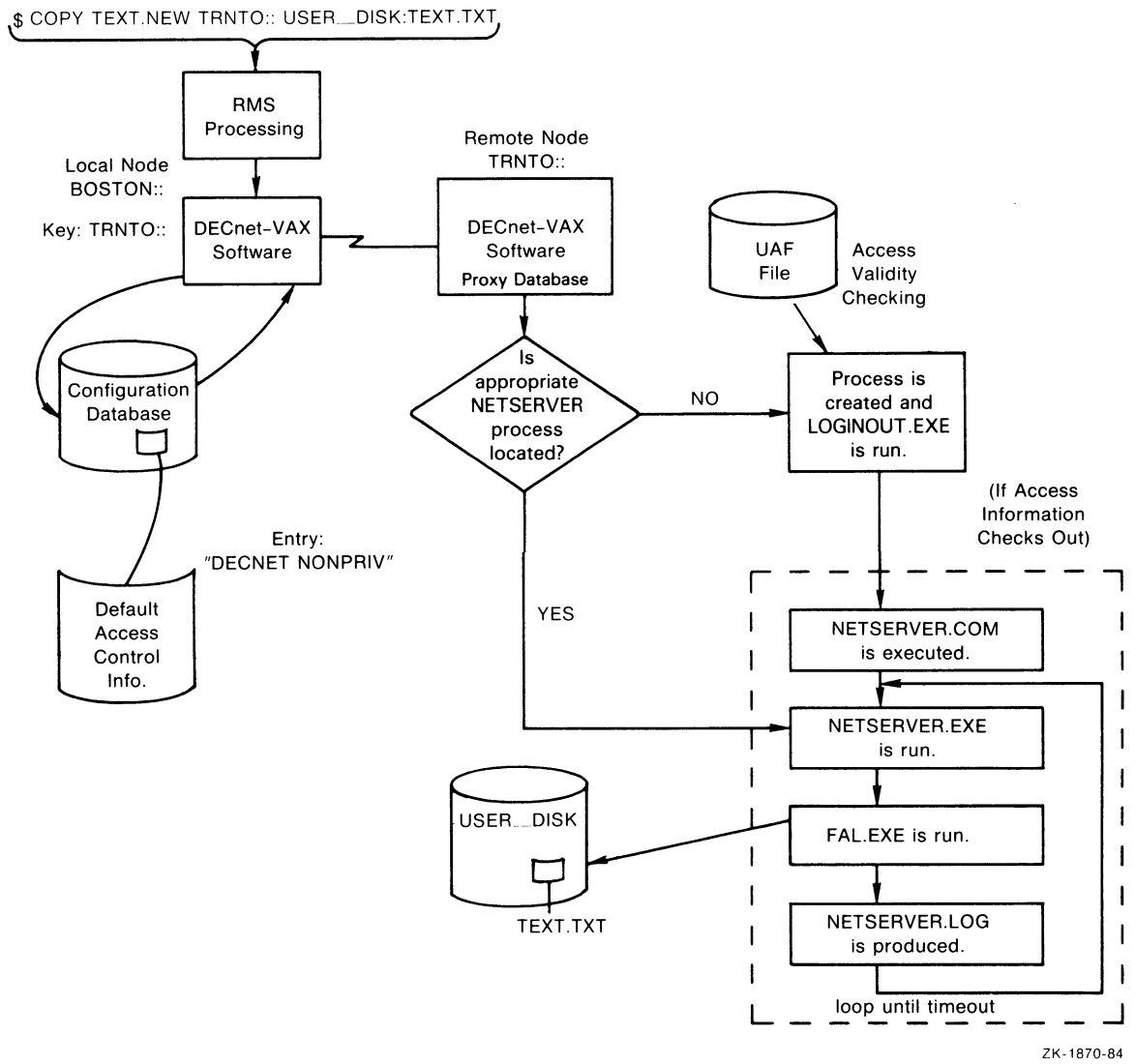
1.5.2.3 Using Logical Names in Network Applications

Using logical names for network operations allows you to refer to network file and task specifications without using actual names that you give these elements. Logical names serve as a kind of shorthand for specifying all or a portion of a full file specification. By using logical names, you can pass file specifications defined at the DCL level to an executing image at run time. For example, logical names allow a program to access local or remote files without changing the program. You can also use logical names to conceal access control information from other users by embedding it in a logical name defined in the process logical name table. Logical names provide convenient and powerful multilevel access control specification.

The rules that govern the use of logical names for network operations are as follows:

- Both the device name and node name in a full file specification string can be logical names. After a node specification is encountered during file parsing, however, a device name that follows is not translated locally. Instead, it is passed unaltered to the remote node, where it is subject to logical name translation.
- A logical name appearing in the device name position in a file specification can supply any file specification string elements when translated.
- A logical name appearing in the node name position can supply only a node specification when translated. Therefore, its equivalence string must end with a double colon.
- An access control string associated with a logical node name becomes the new access control string for the node-specification of the equivalence string, even if the node specification contained an access control string. Thus, you can easily specify a default (or override any) access control string defined for the node specification resulting from logical name translation.
- After a logical node name is translated, the new node name becomes a candidate for logical node name translation.
- A maximum of ten logical device name translations and ten logical node name translations is permitted. If you exceed these limits, an error message is returned.
- While logical name translation is not done on the local node, merging the default name string (and related names) is accomplished locally.

Figure 1–11 Remote File Access Using Default Access Control Information



For more information about logical names, including examples of logical names that can be used for network applications, see the *VMS DCL Concepts Manual*.

2 DECnet–VAX Components and Concepts

This chapter presents networking concepts relevant to understanding the operation of the DECnet network, in terms of the functions performed by DECnet–VAX components and VAX PSI components.

To establish your VMS operating system as part of the DECnet network, you must build and maintain a network configuration database, consisting of records that describe the specific network components your particular system requires. This chapter describes the DECnet–VAX components and their characteristics: nodes, circuits, lines, routing, logical links, objects, logging, and network access control. It also describes VAX PSI components used in communicating over a packet switching data network (PSDN): the X.25 protocol module, X.25/X.29 server modules, and X.25 access module.

Chapter 3 discusses how you can use a DECnet–VAX utility program, the Network Control Program (NCP), to enter in your configuration database specific parameters for each network component your system will use.

2.1 Nodes and DTEs

A node is an operating system that uses DECnet software to communicate with other operating systems across a network. A VMS node uses DECnet–VAX software to communicate with other DECnet–VAX nodes and with any other DIGITAL operating system that supports DECnet.

The X.25 equivalent of a node is the DTE (data terminal equipment). The DTE is a computer or terminal that uses VAX PSI software to communicate with remote nodes across PSDNs. You can configure your DECnet–VAX node as a DTE, provided that VAX PSI software is installed on your node. Note that a single VAX PSI system can support several DTEs. You can connect these DTEs to any combination of one or more PSDNs.

A DECnet–VAX node can also be an X.25 connector node, serving as a gateway that permits DECnet–VAX host nodes on an Ethernet to access one or more PSDNs. To configure your DECnet–VAX node as a connector node, you must have VAX PSI software in multihost mode installed. To configure your DECnet–VAX node as an X.25 host node, you must have VAX PSI Access software installed.

This section describes the characteristics of nodes and the kinds of parameters you can associate with them. It also describes DTEs and indicates how you use X.25 protocol modules to define DTEs and the parameters related to them. Chapter 3 discusses how to use NCP commands to establish the parameters for nodes and DTEs.

2.1.1 Nodes

The VMS operating system at which you are physically located is called the local node. By issuing network management commands at your local node, you can perform configuration, control, and monitoring functions that affect both the local node and other nodes in the network. The node on which network management functions are actually performed is called the executor node. Usually, the executor node is the local node. You have the option, however, of entering at the local node one or more commands to be executed at a remote node. For those commands, the remote node serves as the executor node.

2.1.1.1 Node Address and Name

To configure an operational network at the local node, you must establish configuration database entries for the local node and for all adjacent nodes that are connected by circuits. You should specify names and addresses for all nodes in the network. After you have done so, you can reach any other node by its name.

To satisfy routing requirements, each node in the network must have a unique address. The **node address** is a number in the format:

area-number.node-number

where:

area-number	Is the number of the area in which the node resides.
node-number	Is the address of the node within that area.

Each area number must be unique within the network and each node number unique within the area. If you do not specify the area number in a node address, the area number of a remote node defaults to the area number of the executor node, and the area number of the executor defaults to the number 1.

Node identification has two forms: a node address and a **node name**. A node address, a number in the format described previously, is assigned to each node in the configuration database. A node name is an optional alphanumeric string. In the single-area network example in Chapter 1, the node assigned node address 1.11 is also identified by the node name BOSTON. In the multiple-area network example in Chapter 1, node BOSTON in area 2 has the node address 2.11.

Because it is often easier to remember a name rather than an address, you may prefer to associate a name with an address. You can do so at any time. Note, however, that node names are known only to the local node network software while node addresses are known network-wide by the routing function. To avoid potential confusion, you should give each node a unique name that all nodes in the network will assign to that node and use to address it.

Nodes on Ethernet lines can also be accessed under certain circumstances by their Ethernet addresses. All nodes connected to an Ethernet line are equally accessible, because the Ethernet is a multiaccess, broadcast device. Therefore, each node on an Ethernet is assigned a unique Ethernet physical address, which is set by the software at the node. You do not normally have to specify the Ethernet address of an individual node to configure your network or perform normal network operations. You do need to know a node's Ethernet physical address for service functions (such as downline load, circuit loopback test, and configurator operations). You can send a message

to one, several, or all nodes on an Ethernet line simultaneously, depending on the Ethernet address used. To send a message to more than one node, use an Ethernet multicast address: either a multicast group address to reach a selected group of nodes, or a broadcast address to reach all nodes on the Ethernet.

2.1.1.2 Node Characteristics

The configuration database for the local node must contain certain information about the local node and may contain node information for all nodes with which you want to communicate. For the local node, you must specify the node address and should specify the node name and buffer size (which determines the largest size message the node can forward). You should also indicate or use the default value for the highest address the local node will recognize, and for the node type (which determines the routing capabilities of the local node). Optionally, you can specify data link control information for the local node. For remote nodes, you should specify names and addresses. You can also specify default information to be used in performing downline load or upline dump operations involving remote nodes. For any or all nodes, you can specify access control information and node counter event logging information.

To update your configuration database with current information about remote nodes in your network, you can copy the names and addresses of remote nodes from the database of another node to which you have access. You can specify the node database (volatile or permanent) to be copied, and the local node database (either or both volatile and permanent) to which information is to be copied. You also have the option of clearing or purging your local node database before copying the remote node data, thus avoiding possible conflicts between original and updated data. The executor node information is preserved during the clear or purge operation. Being able to copy a node database permits you to keep your network information current even if you are part of a large network that changes frequently. Alternatively, if you configure your node without a permanent node database, you could obtain current information on other nodes in the network by copying it from another node (for example, from a node on your Ethernet that serves as a master by keeping its node database up to date).

The data link control information you can specify for the local node controls certain characteristics of physical line operation, including the size and number of transmit and receive buffers and the number of circuits the local node can use. You should set these values to levels that ensure reasonable system operation. You must set the buffers for all nodes in the network to the same size. Otherwise, packets will be dropped when routed through nodes with smaller buffer sizes. A procedure for changing the size of buffers on all nodes in the network without bringing down the whole network is given in Section 3.3.5.1.

You can control the operational **state** of the local node and thereby control its active participation in the network. This control is usually a function of whether inbound logical link connections can be established or maintained with the local node. You can use this control to restrict the operation of the node or to shut it down altogether.

2.1.1.3 Identifying a VAXcluster as a Single Node

A whole VAXcluster or some of the nodes in a VAXcluster can be represented by a special identifier called the alias node identifier, which appears to other nodes in the network to identify an actual node. This mechanism allows users on DECnet nodes outside the cluster to access cluster resources without knowing what the cluster nodes are or which are active.

Any node in the cluster can elect to assume the alias node identifier while retaining its own unique node name and address. Use of the alias never precludes use of the individual node name and address. Thus, a remote node can address the cluster as a single node, and address any cluster member individually.

You can designate that your cluster node is assuming the alias node identifier by specifying in your configuration database either the alias node address or the alias node name (if you have previously associated that name with the alias address of the cluster).

You then have the option of indicating whether you want to use the alias for incoming and selected outgoing connections. Specifically, you can indicate whether your node will accept incoming connection requests directed to the alias node address. By default, a node that assumes the alias is available to receive incoming connections addressed to the alias, but a small node that uses the alias for outgoing traffic may elect not to handle the extra incoming traffic. You can also select which DECnet–VAX objects (software components that provide network services) are to use the alias by specifying in the object database that the alias address is to be used for outgoing connections originated by those objects. In addition, you can specify which objects will receive incoming connect requests directed to the alias node address.

MAIL is an example of a network object that can effectively treat the cluster as a single node. Ordinarily, replies to mail messages are directed to the node that originated the message; the reply is not delivered if that node is not available. If the node is in a cluster and uses the cluster alias, an outgoing mail message is identified by the alias node address rather than the individual address of the originating node. An incoming reply directed to the alias address is given to any active node in the cluster and is delivered to the originator's mail file.

Objects that involve multiple incoming links (such as PHONE) should not use the alias node address because each incoming link may be routed to a different node that uses the same alias. Also, objects whose resources are not accessible clusterwide should not be allowed to receive incoming connect requests directed to the alias node address. Section 2.6 describes network objects and discusses the type of object for which the alias node identifier is suitable.

The alias node identifier permits you to set a proxy to a remote node for the whole cluster rather than for each node in the cluster. The clusterwide proxy can be useful if the alias node address is used for outgoing connections originated by the object FAL, which accesses the file system.

You should use the alias node identifier only for fully shareable resources in a VAXcluster. All processors in the cluster must be able to access and share all resources (such as files and devices). Nodes that assume the alias node identifier should have a common authorization file.

At least one of the cluster nodes that uses the alias node identifier must be a router. It can be a level 1 router, because all cluster nodes sharing the same alias node address must be in the same area. The cluster router informs other nodes in the network of the existence of the alias node address. Other routers in the network perceive the cluster router as the shortest path to the cluster node address and send the router packets addressed to the cluster node address. If the router receives a packet addressed to the alias node address, it forwards the packet to the appropriate cluster node: if the packet is for an existing logical link, the link identifier in the packet is sufficient to select the node; if the packet is initiating a new logical link, the router selects a participating node in round-robin fashion.

The network manager or cluster manager should select a suitable alias node name and address for the cluster nodes. You can specify either the alias node name or address as an executor parameter in your node database. If you specify the alias node name, you must first have associated the name with the agreed-upon alias node address. You can then assign the same parameters to this node as to other nodes, except that routing initialization passwords are not required. No point-to-point initialization can occur because a node cannot set up a circuit to an alias node address. The alias node address and name appear in the node databases of other nodes in the network.

You can optionally set a maximum value on the number of logical links that your node can initiate using the alias node identifier (see Section 2.5).

2.1.2 DTEs

A VMS operating system with VAX PSI software installed can function as a DTE (or DTEs) capable of sending and receiving packets over PSDNs to which the system is connected. To configure a local DTE, you must establish in the configuration database an X.25 protocol module entry, which identifies a network to which your DTE is connected. This network represents a path to a PSDN. You must also establish X.25/X.29 server database entries that indicate the destinations of incoming X.25 and X.29 calls (see Section 2.7).

2.1.2.1 X.25 Protocol Module

The X.25 protocol module is a software component that controls the transmission of data packets over PSDNs. The configuration database for the X.25 protocol module identifies the networks to which your DTEs are connected, defines the DTEs, and specifies any user group to be associated with the DTEs.

The first step in configuring the X.25 protocol module is to identify the particular networks with which your DTE or DTEs will connect. Note that the term "network" as used here refers to a network you define with NCP. In this context, network does not refer to a PSDN, but rather a route to a PSDN. You must associate the network name with a profile that determines the characteristics of subscription to the PSDN. When you specify a network profile, default values for a number of parameters that affect data-packet control are entered in the configuration database. Defaults set the size and control the flow of data packets over switched virtual circuits, and control call setup and clearing of these circuits; they also control the transmission of resets and restarts over permanent and switched virtual circuits.

You identify your local DTE or DTEs by DTE address and network name. Each local DTE must have a unique address for the network to which it belongs. The DTE address format is determined by the PSDN whose profile is specified for the network. For each DTE, you can specify the operational state and maximum number of circuits supported, and identify the associated line and the channels for outgoing calls.

You should also identify any user groups of which you are a member. A user group is an optional PSDN facility to which you can subscribe: a **closed user group (CUG)** permits two or more DTEs to communicate only with each other; a **bilateral closed user group (BCUG)** restricts communication to a pair of DTEs. You specify the unique name of your CUG or BCUG and associate with it the local DTE address and group number; for a BCUG, you specify that the group type is bilateral.

2.1.2.2 X.25 Connector and Host Nodes

A VMS node that has VAX PSI software in multihost mode installed can serve as a connector node, a gateway that provides access to a PSDN for host nodes. To configure your node with VAX PSI software in multihost mode, establish in the database an X.25 protocol module entry as described in Section 2.1.2.1, then indicate in the X.25 server database the host destinations to which incoming calls are to be forwarded.

A VMS host node must have VAX PSI Access software installed. To configure your node as a host node, you must establish in the database the X.25 access module (see Section 2.8) and indicate in the X.25 server database the destination on your node for incoming calls.

2.2 Circuits

Circuits are high-level communications data paths between nodes or DTEs; communication between nodes takes place over circuits. Circuits operate over physical lines, which are low-level communications paths (see Section 3.6).

2.2.1 Classes of DECnet–VAX Circuits

DECnet–VAX employs four classes of circuit: DDCMP, CI, Ethernet, and X.25.

DDCMP circuits provide the logical point-to-point or multipoint connection between two or more nodes. There are currently three types of DDCMP circuit: point-to-point, multipoint control, and multipoint tributary. A point-to-point circuit operates over a corresponding synchronous or asynchronous DDCMP point-to-point line. Asynchronous lines can be either static (permanent) or dynamic (switched).

Multipoint control circuits operate over synchronous DDCMP control lines. You can specify multiple circuits from the control (master) end of a control line, but each circuit must have a unique physical tributary address. On the tributary (slave) end, you can specify only one multipoint tributary circuit per line.

CI circuits are available only on those nodes that are attached to a CI–750, CI–780, CIBCA, or CIBCI interconnect. The setup of CI circuits is similar in many ways to the setup of DDCMP multipoint circuits. CI circuits, however, use their own protocol.

Ethernet circuits provide for multiaccess connection between a number of nodes on the same broadcast circuit. An Ethernet circuit differs from other DECnet circuits in that there is not a single node at the other end. An Ethernet circuit is a path to many nodes. Each node on a single Ethernet circuit is considered adjacent to every other node on the circuit and equally accessible. Every node must have a unique node identification: an Ethernet physical address. (Ethernet node addressing is described in Section 3.3.4.) Ethernet circuits use the Ethernet protocol.

X.25 circuits use the X.25 level 3 protocol (the packet level) and provide for communication over PSDNs. VAX PSI provides X.25 circuits for use by PSI user application programs (also referred to as "native X.25 user programs") or by DECnet data link mapping (DLM). DLM permits the use of X.25 as a DECnet data link through the mapping of data link information between the DECnet Routing layer and the X.25 protocol module. The two types of X.25 circuit are X.25 native circuits and X.25 DLM circuits.

Each X.25 circuit is a virtual circuit connecting a local DTE and a remote DTE. An X.25 virtual circuit can be either of the following:

- A permanent virtual circuit (PVC), providing a permanent path between the local DTE and the remote DTE
- A switched virtual circuit (SVC), providing a temporary path between the local DTE and the remote DTE

You need to set up all PVCs (whether used by DECnet or native X.25 user programs) using NCP commands. Also, you must set up SVCs used by DECnet by means of NCP commands. VAX PSI sets up X.25 native SVCs with parameters taken from the X.25 protocol module component when calls on these circuits are requested. You do not need to use NCP to set up X.25 native SVCs.

Just as you must specify the local node, you must also specify parameters for all DECnet circuits connected to the local node and all X.25 virtual circuits connected to local DTEs.

You must identify each circuit by name and specify information that directly affects the circuit's operation. You can also specify the operational state of circuits connected to your local node or DTE. Thus you can control circuit traffic and perform service functions. The state of a circuit may ultimately affect the system's ability to reach an adjacent node or DTE. The circuit state can have a similar effect on routing.

The following sections describe the circuit component. For a discussion of using NCP commands to specify circuits, see Chapter 3.

2.2.2 DDCMP Circuit Devices

DDCMP circuit devices can be synchronous or asynchronous. DECnet–VAX supports the following synchronous DDCMP circuit devices.

Device	Mnemonic
DMB32	DMB
DMC11	DMC
DMR11	DMC
DMP11	DMP
DMV11	DMP
DMF32	DMF

The DMC11 and the DMR11 are point-to-point circuit devices and are considered identical. The DMP11 can be a point-to-point, multipoint control, or multipoint tributary circuit device. The DMV11 is similar to the DMP11; DECnet refers to both devices as the DMP11. The DMF32 and DMB32 synchronous line units are either point-to-point or multipoint tributary circuit devices.

DECnet–VAX supports the following asynchronous DDCMP circuit devices.

Device	Mnemonic
DHQ11	TX
DHU11	TX
DHV11	TX
DMB32	TX
DMF32	TX
DMZ32	TX
DZ11	TT
DZ32	TT
DZQ11	TT
DZV11	TT

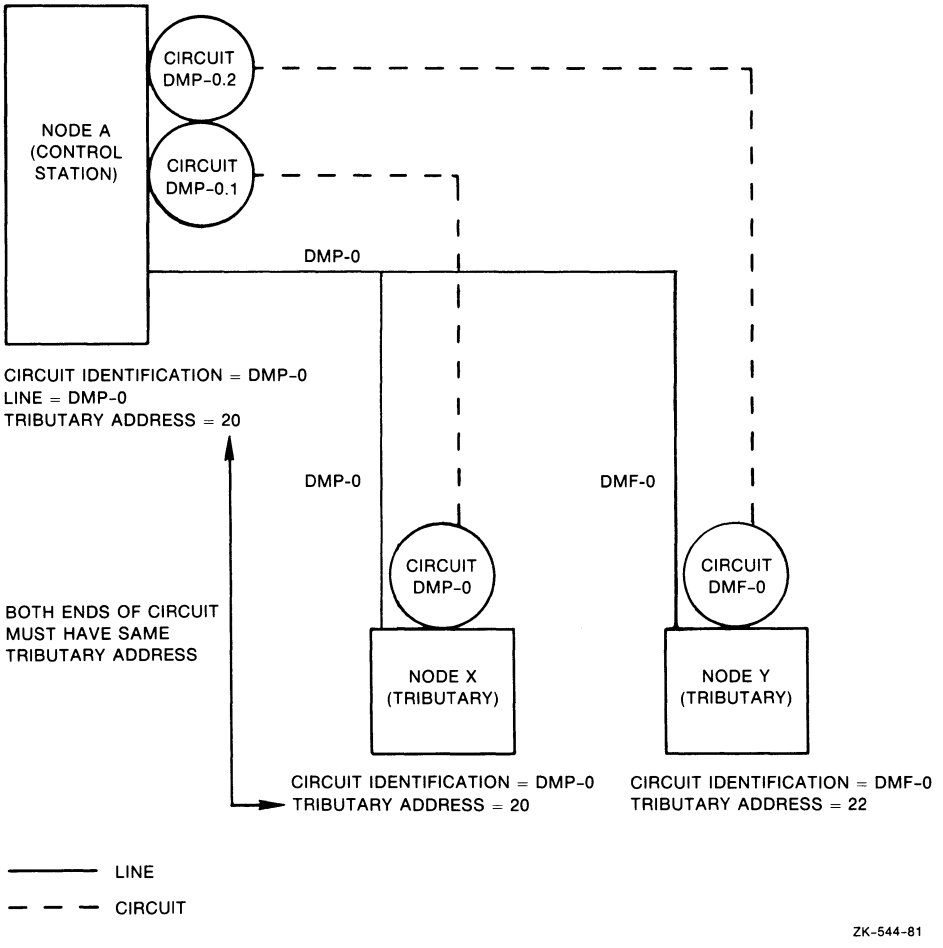
The asynchronous circuit devices are point-to-point circuit devices used for static or dynamic asynchronous connections.

Note that asynchronous DDCMP circuits need not be predefined for dynamic connections. They are established automatically during dynamic switching of terminal lines (see Section 2.3.2.3).

Other DECnet implementations may support other DDCMP circuit devices. If a node in your network uses a circuit device other than one of these, refer to the appropriate DECnet documentation. This section provides a general discussion of point-to-point and multipoint circuits. The *VMS Network Control Program Manual* lists DECnet circuit and line devices by name and operational category.

Every DDCMP circuit provides a logical point-to-point connection between two nodes. The circuit operates over the corresponding DDCMP line (for example, the DMC11 circuit operates over the DMC11 line). The DMP11, operating as a multipoint control circuit, also provides a logical, multipoint connection (over one physical line) between a control station and several tributaries (as illustrated in Figure 2–1). The DMP11 and DMF32 can also operate as multipoint tributary circuit devices that provide a logical connection between a tributary and a control station.

Figure 2–1 Multipoint Circuits and Associated Lines



The following terms are used to describe the operation of multipoint circuits:

- **Control Station**—the node at the controlling end of a multipoint circuit. It controls the tributaries for that circuit.

- **Polling**—the activity that the control station performs on tributaries of a multipoint circuit. The control station regularly sends request messages to (that is, polls) each eligible tributary in the polling list. The request message asks the tributary if it has anything to send (essentially giving it permission to use the bus).
- **Tributary**—a physical termination point on a multipoint circuit that is not a control station.
- **Tributary Address**—a numeric address that identifies a tributary node on a multipoint circuit.

You can connect both a multipoint control circuit and a multipoint tributary circuit to the same node. The node could then serve as the control station for one multipoint circuit and as a tributary for another multipoint circuit.

The system manager must supply tributary addresses for a control station to use when polling each tributary in a polling list.

2.2.3 CI Circuit Devices

DECnet supports the CI–750 interconnect (on VAX–11/750 processors only), the CI–780 interconnect (on VAX–11/780, VAX–11/785, VAX 8600, VAX 8650 processors only), and the CIBCA and CIBCI (on VAX 8200, VAX 8250, VAX 8300, VAX 8350, VAX 8500, VAX 8530, VAX 8550, VAX 8700 and VAX 8800 processors only). All nodes connected to the same CI bus can communicate directly with each other. Only one CI controller per node is required. DECnet treats the CI controller as a multipoint data link and requires a single entry in the line database and multiple entries in the circuit database. The line database entry describes the CI controller (see Section 3.6). Each circuit database entry describes a virtual connection to a single remote node on the CI. CI multipoint circuits and DDCMP multipoint circuits differ in the following ways:

- Each station on the CI can talk directly to every other station. These stations are called tributaries and all stations are alike. There are no "control" and "tributary" stations as with DDCMP multipoint circuits. Only the setup of CI circuits is similar to multipoint circuits.
- There are no polling parameters on the CI.
- CI circuits use their own communication protocol.

If you plan to use a CI circuit, you must first connect the device CNA0 to the driver CNDRIVER. To connect CNA0 to the CNDRIVER and load the CNDRIVER, add the following lines to the LOADNET.COM command procedure in SYS$MANAGER:

```
$ RUN SYS$SYSTEM:SYSGEN
CONNECT CNA0/NOADAPTER
```

2.2.4 Ethernet Circuit Device

DECnet supports the following circuit devices, which provide for multiaccess connections between many nodes on the same Ethernet circuit.

Device	Mnemonic	Bus Name
DEUNA	UNA	UNIBUS
DELUA	UNA	UNIBUS
DEQNA	QNA	Q–bus
DELQA	QNA	Q–bus
DEBNA	BNA	BI–bus
DESVA	SVA	None [1]

[1] The VAXstation 2000 and the MicroVAX 2000 processors use the DESVA.

VMS operating systems configured with a UNIBUS may use the DEUNA or DELUA circuit device (each of these devices is called the UNA). The DELUA is a newer version of the DEUNA that provides higher throughput. VMS operating systems with a Q–bus running on MicroVAXes use the DEQNA and the DELQA (each of these devices is referred to as the QNA), which are similar in function to the DEUNA.

VMS operating systems with a BI–bus may use the DEBNA (referred to as the BNA). The VAXstation 2000 and the MicroVAX 2000 processors use the DESVA (referred to as the SVA). The DWBUA device is a UNIBUS adapter that allows a DELUA or a DEUNA to be connected to a BI system.

All these devices use the Ethernet protocol. Ethernet messages are sent over the Ethernet as datagrams, which means messages may be lost because of errors. DECnet provides for automatic retransmission of lost messages. The Ethernet device allows multiple users of the device at the same time. Therefore, other users may be using the Ethernet device with another protocol type, while DECnet is running.

2.2.5 Ethernet Configurator Module

All nodes on an Ethernet circuit are logically adjacent, because the Ethernet is a multiaccess device. To obtain a list of all systems on an Ethernet circuit, you can use the Ethernet configurator module. The configurator module listens to system identification messages transmitted periodically by every DIGITAL-supported node on the Ethernet circuit, and builds the configuration list from the received messages.

Approximately once every 10 minutes, each node on an Ethernet circuit that conforms to the DNA specifications transmits a system identification message (a hello message) to a multicast address that the configurator monitors. For a random distribution of nodes with possible loss of system identification datagrams, the configurator would require 40 minutes to collect all node addresses. In practice, the configurator normally requires about 12 minutes to complete a list.

The Ethernet configurator module requires a default nonprivileged DECnet account. You use NCP commands to access and control the configurator module. The configurator runs as a separate process and, once it is started, becomes available to all users on the system. The configurator module continues to execute and maintains and updates its database of information on active nodes.

When you request information about the current configuration of nodes on Ethernet circuits, the following is displayed for each system: its Ethernet physical and hardware addresses, the device connecting it to the circuit, maintenance functions it can perform, and the time of the last system identification message from the system.

2.2.6 X.25 Circuit Devices

X.25 circuits differ from DDCMP circuits in that there is no direct correspondence between circuit and line. All X.25 circuits pass through the X.25 protocol handler module (see Section 2.1.2.1), which multiplexes circuits to lines that it "owns." There is no direct relationship between the name of an X.25 circuit and an X.25 line. One line is specified for each DTE.

All X.25 circuits are virtual circuits that connect a local DTE with a remote DTE. The association between DTEs can be permanent or temporary. X.25 PVCs are analogous to leased lines between the local DTE and the remote DTE. They are similar to DDCMP circuits in that both have predefined end points.

X.25 SVCs are analogous to dialup lines. You set up SVCs only when there is data to transmit; SVCs are cleared when the transfer is complete. They are temporary paths between local and remote DTEs.

2.2.7 X.25 DLM Circuits

Data link mapping (DLM) circuits extend normal DECnet capabilities to include communication over a PSDN with other DECnet nodes connected to the PSDN. Data link mapping permits an X.25 virtual circuit to be used as a DECnet data link. A DLM circuit is owned by the executor node; the Routing layer has exclusive rights to use the circuit. A DLM circuit can be either a PVC or an SVC. A DLM SVC can be used for either incoming or outgoing calls.

To establish a DLM SVC with a remote DTE, DECnet–VAX uses the DTE address of the remote node. Subaddresses can be used to limit the DLM calls accepted at the local DTE. DECnet–VAX will try to recall a number if previous attempts to establish a DLM SVC have not succeeded. The number and frequency of recall attempts can be regulated.

2.3 Lines

Lines provide physical communications and are the lowest level communications path. Circuits are high-level communications paths that operate over lines.

2.3.1 Classes of DECnet–VAX Lines

DECnet–VAX supports four classes of line: DDCMP, CI, Ethernet, and X.25. A DDCMP line provides the physical point-to-point or multipoint connection between two or more nodes. A CI line provides a high-speed connection between two or more nodes. An Ethernet line is a multiaccess connection between two or more nodes. An X.25 line is the physical link between your DTE and a PSDN.

For DDCMP, CI, and Ethernet configurations, each circuit is directly related to a corresponding line. For X.25 configurations, however, the circuits and lines do not correspond directly. X.25 circuits are multiplexed to lines owned by the X.25 protocol handler module (see Section 2.1.2.1).

Just as you must establish node and circuit parameters, you must also establish parameters for all physical lines connected to the local node or DTE. You must identify each line by name and specify information that directly affects the line's operation. You can control the operational state of the line, and thus control line traffic and perform service functions. The state of a line may ultimately affect the reachability of an adjacent node or DTE, affecting the routing.

The following sections describe the line component. For a discussion of using NCP commands to specify line parameters, see Chapter 3.

2.3.2 DDCMP Lines

DDCMP lines can be synchronous point-to-point or multipoint lines or asynchronous point-to-point lines. Asynchronous lines can be static (permanent) or dynamic (temporarily switched).

2.3.2.1 DDCMP Line Devices

DDCMP line devices can be synchronous or asynchronous. (For a complete list of DDCMP devices and their corresponding mnemonic names, refer to Section 2.2.2.) DECnet–VAX supports the following synchronous DDCMP line devices:

- DMB32 synchronous line unit
- DMC11
- DMR11
- DMP11
- DMV11
- DMF32 synchronous line unit

The DMC11 and the DMR11 are point-to-point line devices and are considered identical. The DMP11 can be either a point-to-point, multipoint control, or multipoint tributary line device. The DMV11 is similar to the DMP11; DECnet refers to either device as the DMP11. The DMB32 and DMF32 synchronous line units are point-to-point or multipoint tributary line devices.

2.3 Lines

DECnet–VAX supports the following asynchronous DDCMP line devices:

- DHQ11
- DHU11
- DHV11
- DMB32 asynchronous line unit
- DMF32 asynchronous line unit
- DMZ32
- DZ11
- DZ32
- DZQ11
- DZV11

The asynchronous line devices are point-to-point line devices used for static or dynamic asynchronous connections.

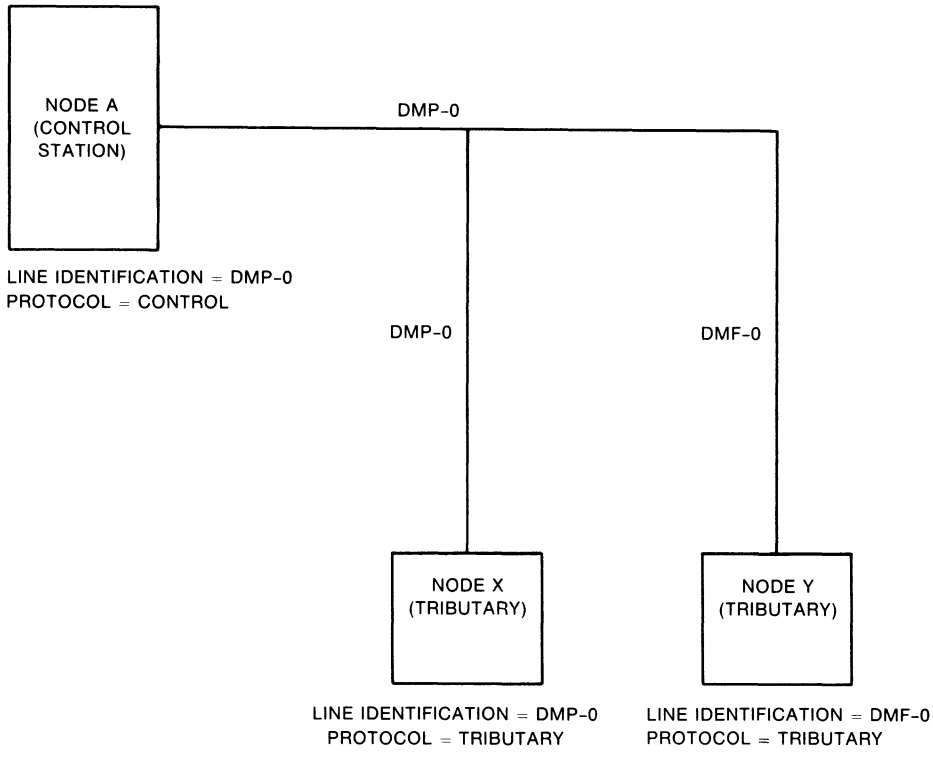
Note that asynchronous DDCMP lines need not be predefined for dynamic connections. They are established automatically when a dynamic asynchronous DDCMP connection is made (see Section 2.3.2.3).

Every DDCMP line provides a point-to-point connection between two nodes. Circuits, the actual communications path, operate over the line. The DMP11 and DMF32 also provide a multipoint connection between two or more nodes. In Figure 2–2 a multipoint line controlled by the DMP11 provides the physical connection between a control node and several tributary nodes.

You can connect two multipoint lines to the same node. The node could then serve as the control station for one multipoint line and as a tributary for another multipoint line.

Because a heterogeneous network may have DDCMP line devices other than one of the preceding, you should be familiar with the entire range of devices and their impact on network management. If a node in your network uses a line device other than these, refer to the appropriate DECnet documentation. The *VMS Network Control Program Manual* lists DECnet line devices by name and operational category.

Figure 2–2 Multipoint Lines



2.3.2.2 Static Asynchronous Lines

A static asynchronous DDCMP connection is a permanent connection established between two nodes (such as a VMS router node and a VMS end node). The two nodes are connected by either a modem or by a physical line attached to a terminal port at each end (for example, port TTA0 on the end node and port TXB7 on the router). A static asynchronous connection can also be made over a dialup line.

Before the DECnet connection is made, the terminal lines must be converted to static asynchronous DDCMP lines. Each terminal port must have an asynchronous DDCMP line device installed, and the system manager at each node must load the asynchronous DDCMP driver, NODRIVER. The system manager at each node should insert the following command in the SYSTARTUP_V5.COM command procedure:

```
$ SET TERMINAL/PROTOCOL=DDCMP device-name:
```

where:

device-name Is the name of the appropriate terminal port.

For example, the manager of a MicroVAX running the VMS operating system should specify the following command:

```
$ SET TERMINAL/PROTOCOL=DDCMP TTA0:
```

The manager of the VMS router should specify this command:

```
$ SET TERMINAL/PROTOCOL=DDCMP TXB7:
```

Each system manager should then specify the appropriate line and circuit commands in the configuration database to turn on the line and circuit for DECnet use. (The commands required to install static asynchronous lines and the NCP commands to configure a network using static asynchronous lines are given in Chapter 5.)

2.3.2.3 Dynamic Asynchronous Lines

A dynamic asynchronous line differs from a static asynchronous line or other DECnet–VAX line in that it is normally switched on for network use only for the duration of a dialup connection between two nodes. When the telephone is hung up, the line reverts to being a terminal line.

Figure 2–3 illustrates a typical configuration in which dynamic asynchronous switching occurs over a dialup line. The local node in Figure 2–3 is a standalone MicroVAX II system; the remote node is a VAX–11/780. After the user at the local node dials in to the remote node, he or she can cause the lines connected to terminal ports TTA1 and TXB1 to be switched to dynamic asynchronous DDCMP lines for use in DECnet communications.

Dynamic switching of terminal lines to asynchronous DDCMP lines can occur provided both nodes have DECnet installed. Assuming that both the remote node and the local node are VMS operating systems, the system manager at each node must have loaded the asynchronous driver NODRIVER and installed the privileged shareable image DYNSWITCH. (If the local node is a personal computer, there is no need to load NODRIVER and install DYNSWITCH.) The system manager at the remote node must have enabled the use of virtual terminals on the system. First, the system manager must have enabled the use of the virtual terminal for the line over which you are going to log in by issuing the CONNECT command of the SYSGEN Utility. The system manager must also have enabled virtual terminals on the terminal line using the DISCONNECT attribute of the SET TERMINAL command for the terminal.

A functional explanation of the procedure for dynamic switching of lines, as shown in Figure 2–3, is as follows:

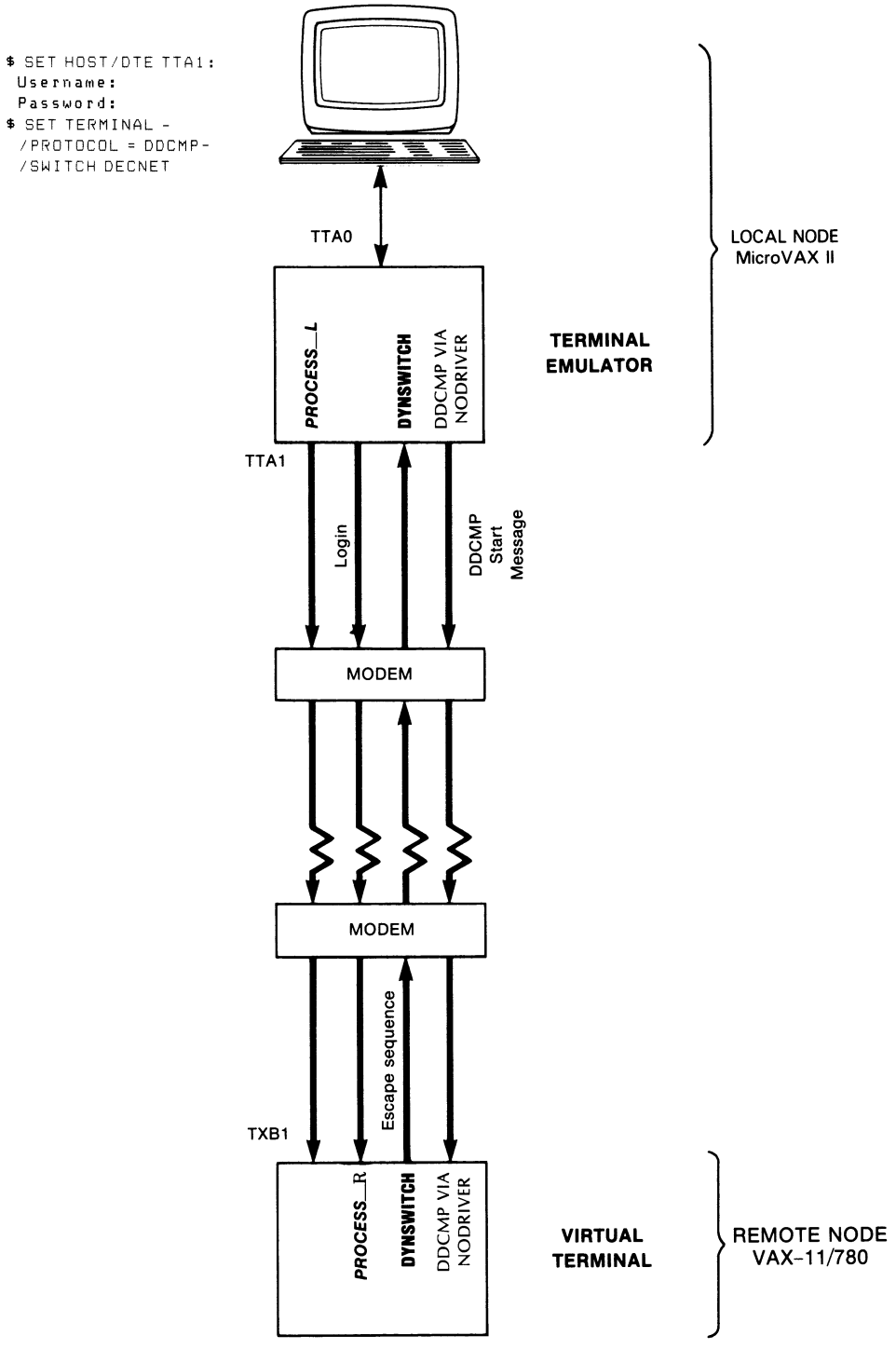
1. You should log in to the VMS operating system running on the MicroVAX II, causing a process to be created on your system. In Figure 2–3, this process is identified by the sample process name PROCESS_L.

2. You must enter the following DCL command:

   ```
   $ SET HOST/DTE[/DIAL=NUMBER:number] TTA1:
   ```

   This command causes a process on the local system to function as a **terminal emulator**, and causes the modem to dial the number of the remote system. The terminal emulator permits the local process to function as though it were a terminal line: characters can be read from one port and written to another port. In Figure 2–3, the terminal emulator on the MicroVAX II reads characters from port TTA1 and writes characters to port TXB1. Note that the /DIAL qualifier in the SET HOST/DTE command is optional and works only if you have written a program to dial your modem. The default program supplied with the VMS operating system dials a DF03 modem.

Figure 2–3 Dynamic Switching of Asynchronous DDCMP Lines



3 If you do not specify the /DIAL qualifier in step 2, dial the remote system manually. After the dialup connection is made and you receive the remote system welcome message, you should perform the regular procedure for logging in to your account on the remote node. In this case, you would supply your user name and password to the remote VMS operating system.

4 When you log in over a modem line, a process is created at the remote node and connected to a **virtual terminal** as well as the physical terminal. In Figure 2–3, this process is identified by the sample process name PROCESS_R. The virtual terminal permits PROCESS_R to continue running even if the physical terminal is disconnected (for example, if you lose the carrier signal on your telephone line).

5 You can then initiate dynamic switching by specifying the following DCL command from your account on the remote node:

```
$ SET TERMINAL/PROTOCOL=DDCMP/SWITCH=DECNET
```

Note that the SET TERMINAL command is a VMS DCL command. If you are on a non-VMS node, you should specify the equivalent function for your system.

6 When the SET image at the remote node recognizes the /SWITCH=DECNET qualifier, it calls the shareable image DYNSWITCH. DYNSWITCH verifies that the device is a virtual terminal and then sends an escape sequence to the terminal emulator running on the MicroVAX II. The escape sequence notifies the terminal emulator that the line connected to the remote terminal port is becoming an asynchronous DDCMP line.

7 When the terminal emulator at the local node receives the escape sequence, it calls the image DYNSWITCH, which causes the line connected to terminal port TTA1 to be switched to an asynchronous DDCMP line. It assigns a channel to the network and supplies the appropriate line and circuit entries to the NCP volatile database at the local node. (Note that the modem line is not dropped; redialing is not required.)

8 The asynchronous DDCMP protocol on the local node sends a DDCMP start message to DYNSWITCH on the remote node and sends the transmit password defined in the local node database. DYNSWITCH at the remote node disconnects the physical terminal from the virtual terminal, and causes the line connected to the physical terminal port (in Figure 2–3, the port TXB1) to be converted to an asynchronous DDCMP line. DYNSWITCH assigns a channel to the network and supplies the appropriate line and circuit parameters to the volatile database to start up the line and circuit.

9 After DECnet is started on the local node, the terminal emulator is exited and control is returned to the local node when the following message is displayed:

```
%REM-S-END, control returned to node _local-node-name::
```

A prompt appears on the local terminal and you can then use DECnet to perform operations over the network.

10 If the terminal emulator does not recognize escape sequences (if the local node is not a VMS operating system), you must specify the /MANUAL qualifier in the SET TERMINAL command indicated in step 5.

```
$ SET TERMINAL/MANUAL/PROTOCOL=DDCMP/SWITCH=DECNET
```

The /MANUAL qualifier prevents DYNSWITCH at the remote node from sending the escape sequence. Instead, DYNSWITCH sends the following message to the local node:

```
%SET-I-SWINPRG, The line you are currently logged in
                over is becoming a DECnet line
```

After receiving this message, if you decide not to switch the line, you can press CTRL/C or CTRL/Y to abort the switch. If your local system is a VAX and you want to continue the switch, you should exit the terminal emulator and switch your terminal line to an asynchronous DDCMP line manually by entering the following command:

```
$ SET TERMINAL/PROTOCOL=DDCMP TTA1:
```

Then, you enter NCP commands to turn on your line and circuit. For example, enter the following commands:

```
NCP> SET CIRCUIT TT-0-1 STATE ON
NCP> SET LINE TT-0-1 STATE ON
```

DYNSWITCH waits 60 seconds for the DDCMP start message and the transmit password and then times out the switch.

Note that the SET TERMINAL command is a VMS DCL command. If you are on a non-VMS node, you should specify the equivalent function for your system.

11 When you hang up the telephone, the line is switched back to a terminal line. (DECnet–VAX automatically clears the line and circuit entries from the volatile database). Alternatively, you can switch the asynchronous line back to a terminal line by issuing an NCP command to turn off the line or circuit.

If you specified the /NOHANGUP qualifier in the SET TERMINAL command in step 5, the modem signal is not dropped if you specify an NCP command to turn off the DECnet line. Therefore, you do not have to redial the connection to the remote node when you want to convert your line to DECnet use.

Chapter 5 provides examples of the commands required to install dynamic asynchronous lines and the NCP commands required to configure a network using dynamic asynchronous lines. The complete DECnet–VAX installation procedure, including establishment of asynchronous connections, appears in the *Guide to DECnet–VAX Networking*.

Chapter 3 describes the NCP command parameters required for asynchronous connections. Section 2.10 summarizes security for dynamic asynchronous connections.

2.3.3 CI Line Device

The CI–780, CI–750, CIBCA, and CIBCI lines are high-speed devices, each of which provides a connection between two or more nodes. If you plan to run DECnet over a CI, you must first connect the device CNA0 to the driver CNDRIVER. To connect CNA0 to the CNDRIVER and load the CNDRIVER, add the following lines to the LOADNET.COM command procedure in SYS$MANAGER:

```
$ RUN SYS$SYSTEM:SYSGEN
CONNECT CNA0/NOADAPTER
```

2.3.4 Ethernet Line Devices

A VMS Ethernet node is connected to the Ethernet line by an Ethernet communications controller, a transceiver, and a transceiver cable. The Ethernet controllers use the Ethernet line protocol. The Ethernet circuit operates over the Ethernet line.

A particular Ethernet node is identified by the Ethernet hardware address of its line device; this hardware address is stored in read-only memory in the Ethernet controller. When DECnet starts an Ethernet line, it constructs an Ethernet physical address for the node (see Section 3.3.4). Shutting off machine power or changing the state of the Ethernet line to OFF causes the Ethernet controller to reset the physical address to the original hardware address. Note that if more than one application will use a particular Ethernet line (for example, DECnet and LAT), DECnet must be brought up first because it resets the physical address.

For a complete list of Ethernet controllers and associated mnemonic names, refer to Section 2.2.4.

2.3.5 X.25 Line Devices

VAX PSI supports the following line devices.

Device	Mnemonic
DMF32 synchronous line unit	DMF
DMB32 synchronous line unit	DMB
DPV11	DPV
DST32	DST
DUP11–DA	DUP
KMS11–B	KMX
KMS1P	KMY
KMV1A	KMV

The DMF32 synchronous line unit is a low-speed synchronous line interface. The DUP11–DA is a low-speed synchronous interface. The DPV11 is equivalent to a DUP11. The combination of the KMS11–B controller hardware and the X.25 level 2 microcode provides a low-speed synchronous interface called the KMX. The KMS11–B supports eight lines, but all lines used must be connected to the same PSDN. Similarly, the combination of the KMS1P controller hardware and the X.25 level 2 microcode provides a medium-speed synchronous interface called the KMY that supports one line. The KMV1A is similar to the KMY. However, the KMV1A provides only a low-speed interface. The DMB32 synchronous unit is a medium-speed, synchronous interface.

For the most recent list of supported X.25 line devices, refer to the VAX P.S.I. Software Product Description (SPD).

2.4 Routing

Routing is the network function that determines the path or route along which data (called "packets" in this context) travels to its destination. The Routing layer of DECnet handles routing functions. Because the need for routing pervades network operation, as much as possible is done in software to relieve you from worrying about the configuration of the network.

As system manager, however, you need to be concerned with the configuration of the network in terms of routing. You must configure each network node as either a routing or a nonrouting node, and you have the option of dividing the whole network into different areas. In addition, certain parameters in the configuration database permit a degree of indirect control over network routing, but, for most networks, the default values of these parameters are reasonable.

For very large networks, it may be helpful to have a network manager oversee the operation of the network as a whole. The network manager could ensure that all node addresses are unique and that routing control parameters provide for efficient data flow through the network.

The following sections explain the different types of routing and nonrouting nodes and configurations, describe the levels of routing, and summarize special routing techniques used with Ethernet. They also introduce basic terms and concepts involved in routing control. Chapter 3 discusses the NCP command parameters that affect routing.

2.4.1 Routing and Nonrouting Nodes

Routing nodes (routers) are nodes that can send and receive packets and route packets from one node to another. Routers have two or more circuits. Routers regularly receive and maintain information about other nodes. They perform the routing operation by associating a circuit with the destination node for the packet and transmitting that packet over that circuit. Routers can use DDCMP, CI, Ethernet, or X.25 circuits as their data links.

In a multiple-area network, all routers in a particular area can route packets within the area; some of these routers can also route packets to and from other areas. The two kinds of routers used in area routing configurations are level 1 routers and level 2 routers.

The level 1 router performs intra-area routing within a single area of the network. Note that if all nodes are configured in the same area, the whole network is considered a single area, and all routers are level 1 routers. The level 2 router performs intra-area routing within its own area and interarea routing between its area and one or more other areas of the network.

Nonrouting nodes (end nodes) contain a subset of network software that permits them to send packets or receive packets addressed to them, but not to route packets to other nodes. End nodes have a single circuit connecting them to the rest of the network. They do not send or receive information about network configurations. If two end nodes are connected by a nonbroadcast circuit, these nodes constitute the entire network.

On an Ethernet, if there are two or more routers, one router is elected the designated router to provide message routing services for end nodes on the Ethernet. If no routers are available, Ethernet end nodes can communicate with each other directly by sending a packet out over the Ethernet and then waiting until the timeout for a reply. However, routers are the only Ethernet nodes that can route messages to network nodes not on the Ethernet.

2.4.1.1 Types of DECnet Nodes

DECnet supports a variety of types of nodes developed during different phases of DNA implementation. Phase II, III, and IV nodes can all exist on a network and can be configured adjacently, as follows:

- Phase II/Phase II
- Phase II/Phase III
- Phase III/Phase III
- Phase III/Phase IV
- Phase IV/Phase IV

Phase II nodes can communicate with each other as long as there is a physical data link between them. They support only point-to-point connections. There is no Phase II support for Ethernet.

Phase III DECnet introduced adaptive routing, which allows a reasonably large number of nodes to communicate conveniently. A network composed of Phase III nodes is limited to a practical size of approximately 100 nodes, because of the overhead of routing update messages that have to be exchanged among routers. (The design limit for the address of a DECnet–VAX Phase III node is 255; this limit may vary for other DECnet Phase III systems.) Phase III introduced routers and end nodes. Phase III depends on data links that guarantee delivery of messages in order to accomplish initialization among routers. Therefore, there is no Phase III support for Ethernet.

Phase IV DECnet permits the configuration of very large networks and expands the types of data links available for use. Phase IV supports area routing, which allows configuration of a network of up to 63 areas, each containing up to 1023 nodes. Phase IV software supports Ethernet circuits, as well as DDCMP, CI, and X.25 circuits.

Phase IV nodes can communicate with Phase III nodes. Certain restrictions apply, however, in a mixed Phase III/Phase IV network:

- A Phase III node should not be included in a path between Phase IV nodes.
- A Phase III node in a Phase IV multiple-area network should not be linked with nodes outside its own area.
- Routing initialization passwords (described in Section 2.10.1) are required when a Phase III node is initialized in a Phase IV network.

Section A.5 discusses restrictions on the use of Phase III nodes in Phase IV networks.

2.4.1.2 DECnet–VAX Phase IV Nodes

DECnet–VAX Phase IV nodes are either of the following two types:

- Phase IV routers. These nodes deliver packets to and receive packets from other nodes, and route packets from other source nodes through to other destination nodes. They use Ethernet, DDCMP, X.25, and CI circuits. In an area network configuration, Phase IV routers exist at two routing levels:
  - The level 1 router, which performs routing within a single area. The node type is ROUTING IV.
  - The level 2 router, which performs routing within its own area and to and from other areas. The node type is AREA.
- Phase IV nonrouting nodes (end nodes). These nodes deliver packets to other nodes and receive packets from other nodes, but do not route packets through. They can be attached to an Ethernet, DDCMP, X.25, or CI circuit. The node type is NONROUTING IV.

DECnet–VAX Phase IV nodes can also communicate with the other types of node supported by DECnet. Area numbers are dropped when a Phase IV node communicates with a node that is not a Phase IV node. A Phase IV node adds its executor area number to the node address of a message that it receives from a Phase III node. Nodes with which Phase IV DECnet–VAX nodes can communicate include the following:

- Phase III routers. These nodes deliver packets to and receive packets from other nodes, and route packets from other source nodes through to other destination nodes whose addresses are less than 256. They use DDCMP, X.25, and CI circuits, but do not support Ethernet circuits.
- Phase III nonrouting nodes (end nodes). These nodes send packets to other nodes and receive packets from other nodes, but do not route packets through. These nodes cannot support the Ethernet. DECnet–VAX never provided this type of node, but can communicate with Phase III end nodes (for example, RSX Phase III end nodes).
- Phase II nodes. These nodes can send packets to adjacent Phase III routers or to other adjacent Phase II nodes. However, Phase II nodes can send packets only in point-to-point configurations. In addition, a Phase III node cannot communicate with a Phase II node through another Phase III node.

2.4.1.3 Routing Features of DECnet–VAX License Options

The DECnet–VAX license permits you to use either of two kinds of DECnet–VAX capability:

- Full function
- End node

The full function license permits the use of both DECnet–VAX routing and end node capabilities. The end node license permits a node to be used only as an end node. An upgrade from end node to full function capabilities is available.

Both licenses permit the use of any kind of data link (DDCMP, CI, Ethernet, X.25). Section 6.1 describes how the DECnet–VAX licenses are enabled to turn on the appropriate capability.

A configuration consisting only of end nodes offers certain advantages:

- Less use of the central processor is required for routing.
- Data link efficiency is increased: there is no routing overhead and no route-through traffic occurs over the circuit.

End nodes also involve the following limitations:

- The user on an end node cannot directly see the status of other nodes in the network, because end nodes rely on routing nodes to maintain that information. However, an end node can communicate with other nodes in the network, including nodes outside its own area.
- At most, only one circuit is allowed to be active. If that one link to the network fails, no alternative connection is available until the system manager turns on a standby data link, if one exists.

2.4.2 Area Routing

Phase IV DECnet permits implementation of very large networks through the use of area routing techniques, while still supporting configuration of smaller networks that are not divided into areas. The network manager has the option of partitioning a large network into areas. Each area is a group of nodes. Nodes are grouped together in areas for hierarchical routing purposes. Hierarchical routing involves the addition of a second level of routing to the network. Routing within an area is referred to as level 1 routing; routing between areas is called level 2 routing.

Area routing offers the following advantages:

- Permits configuration of very large networks of more than 1023 nodes.
- Requires less routing traffic, restricting routing overhead between areas to the level 2 routers. Level 1 routers exchange routing information only about nodes in their own area.
- Allows different organizations to manage their nodes separately within a large network.
- Makes the merging of existing networks easier.

When a level 1 router receives a packet destined for a node in another area, it uses level 1 routing to send the packet to the nearest node within its own area that can perform level 2 routing. That router forwards the packet by level 2 routing to a level 2 router in the destination area, which in turn sends the packet by level 1 routing to the destination node in its area.

Note that, if two or more level 2 routers exist in the same area, each level 1 router in that area sends packets destined for other areas to the nearest level 2 router, regardless of which level 2 router is closest to the destination area. The level 1 router has no access to level 2 routing information.

Each area in the network is assigned an area number. Every node in the area is uniquely identified by the addition of its area number as a prefix (followed by a period) to its node number. For example, node 15 in area 7 is addressed as node 7.15. The node number must be unique within the area, but may be used again within another area. Thus, node identification within an area is independent of node identification within other areas.

Phase IV DECnet permits configuration of a maximum of 63 areas (areas 1 through 63), each containing up to 1023 nodes. A Phase IV node address is a 16-bit number: the most significant six bits define the area number, and the least significant 10 bits specify the node number within the area. You can convert the Phase IV node address to its decimal equivalent for use in commands, such as COPY and MAIL, that do not recognize the area prefix (the conversion procedure is given in Section 3.7.2). You can convert to its hexadecimal equivalent for use in determining the Ethernet physical address of the node (the conversion procedure is given in Section 3.3.4.2).

You assign the node address to your own node when you configure it. If you do not specify the area number when addressing a remote node, that node is assumed to be in the same area as your local node.

In a network not divided into multiple areas, each router performs level 1 routing throughout the network.

The characteristics of level 1 and level 2 routing nodes are described in the following section. Section 2.4.4.3 presents rules for configuring hierarchical networks using area-routing techniques. This appendix also describes the configuration of mixed area networks, involving Phase III and Phase IV nodes, and recommends procedures for converting a nonarea network to an area network.

2.4.3 Level 1 and Level 2 Routers

An area can contain many level 1 routers and end nodes, and must contain at least one level 2 router to provide the connection to other areas. A level 1 router acts as a standard routing node. It keeps information on the state of nodes within its own area. Level 1 routing nodes and end nodes obtain access to nodes in other areas through a level 2 router residing in their own area.

A level 2 router keeps information on the state of nodes in its own area and also information on the **cost** and **hops** involved in reaching other areas. (The logical distance between adjacent level 2 nodes is one hop.) The level 2 router always routes packets over the least cost path to a destination area. Level 2 routers have the following characteristics:

- Level 2 routers connect areas.
- Level 2 routers also act as level 1 routers within their own area.

- Each level 2 router in a network must be physically connected to at least one other level 2 router.
- A level 2 router serves as a level 1 router when it is not physically connected to another level 2 router.
- All level 2 routers must be connected in such a way that they create a network of their own.
- Level 2 routers exchange level 2 routing messages among themselves.
- In any given area, there can be more than one level 2 router.
- Each level 2 router indicates it is the nearest level 2 router to each level 1 node in its own area, but each level 1 node decides what its level 2 router is on the basis of cost.

2.4.4 Ethernet Routers and End Nodes

Two special concepts are involved in routing over an Ethernet circuit: the designated router and end node caching.

2.4.4.1 Ethernet Designated Routers

If there are two or more routers on the same Ethernet, one of them is elected as the designated router. By convention, the router with the highest numerical priority (the router priority parameter is set as a CIRCUIT characteristic in its database) is elected router for the circuit. In case of a tie, the node with the highest address is elected as the designated router. The function of the designated router is to route messages over the Ethernet on behalf of end nodes. A designated router is elected even if there are no end nodes currently on the Ethernet.

Ethernet end nodes can also exchange messages directly without using a router. Routers are needed, however, when messages are to be routed to nodes off the Ethernet.

Ethernet end nodes are informed of the identity of the designated router on that Ethernet. End nodes transmit multicast hello messages, so that routers know of their presence on the Ethernet. End nodes keep no information about the network configuration, except that they are permitted to keep a cache of nodes within their area that they may address directly on the Ethernet, rather than going through a router (see Section 2.4.4.2). Thus, an end node may send a packet directly to another Ethernet end node, if the address has been cached, or it may send a packet to the designated router for forwarding.

Note that end nodes can exist on an Ethernet without a router. When an end node on the Ethernet wants to communicate with another end node, and notes that no designated router exists, it always sends the packet directly to the addressed node. If the addressed node is active, the sender receives a reply; if the addressed node is not available, a timeout occurs.

2.4.4.2 Ethernet End Node Caching

End nodes normally send packets by means of a router. To minimize the space and time overhead involved in the routing function on Ethernet circuits, a caching mechanism is available that takes advantage of the fact that nodes on an Ethernet are logically one hop away from each other (one hop is the distance between two adjacent nodes).

An end node maintains a cache of limited size of the addresses of the target nodes with which it has had contact. When a designated router is present and an end node is ready to send a packet to a specific target node for the first time, the end node sends the packet to the designated router, which in turn forwards the packet to the target node. When there is no designated router on the circuit, the end node sends the packet directly, because it expects that the other node is there. By means of the acknowledgment messages it receives, the end node builds its cache of addresses of specific nodes. If a response is received from the target node, the end node examines the received packet for the existence of specific bits (the bits are checked even if the first packet went to the designated router). If the "on-Ethernet" bit is set, which indicates that the target node is on the same Ethernet as the end node, then the next packet can be sent directly, rather than by means of the designated router. If the received packet has the "intra-Ethernet" bit set (which indicates that the target node is not on the same Ethernet as the end node, but is reachable through a routing node that is on the Ethernet), then the next packet can be sent from the end node to the target node by means of a routing node, rather than by means of the Ethernet's designated router.

In summary, the end node uses the acknowledgment messages it receives to build a cache of addresses of target nodes that either are on the same Ethernet or can be reached through a node on the Ethernet. This mechanism is called *reverse path caching*.

2.4.4.3 Area Routing on an Ethernet

All nodes on an Ethernet need not be in the same area; you can configure more than one area on a single Ethernet. The areas on the same Ethernet are logically separate from each other. When you configure two level 1 routing nodes on an Ethernet in different areas, the nodes do not communicate directly with each other. Each level 1 router communicates with a level 2 router in its own area, which sends the message to a level 2 router in the other area. The level 2 router that receives the message then transmits it to the second level 1 router. Section A.6 illustrates area routing on an Ethernet.

2.4.5 Routers and End Nodes on CI Data Links

You can configure nodes using a CI data link in a VAXcluster as routers or as end nodes.

2.4.5.1 CI End Nodes

You can configure a two-node VAXcluster that uses a CI data link using end nodes only, but at least one router is required if additional nodes are configured in the cluster. The CI protocol does not include the multiaccess capabilities of the Ethernet protocol.

2.4.5.2 CI Routers

One or more CI routers are necessary if a VAXcluster consists of three or more nodes. CI circuit devices are treated as though they were multipoint devices (like the DMP device) rather than as multiaccess devices such as the Ethernet circuit device. Although only one router is required in a cluster of more than two nodes, having more routers in the cluster environment increases the overall availability of the network within the cluster.

If the VAXcluster configuration includes end nodes as well as routers, a backup, higher-cost circuit could be provided for each end node. This backup circuit could take over if the primary circuit connecting the end node to its router fails (see Section 3.7.6).

Note that end nodes communicating through a router send all data through that router even though they are connected to the same CI. You achieve the best performance and availability by defining all VAXcluster nodes as routers if the CI is used as the data link.

2.4.6 Routing Concepts and Terms

This section briefly explains routing concepts and defines those routing parameters that provide some control over network routing. Chapter 3 describes how to use NCP commands to set these routing parameters. A more detailed explanation of routing concepts and the routing algorithms for the routing layer can be found in the *Introduction to DECnet Phase IV* manual.

The following terms are used to describe DECnet routing and routing parameters:

- **Hop.** The logical distance between two nodes is measured in hops. The distance between two adjacent nodes is one hop.
- **Path.** A path is the route a packet takes from source to destination.
- **Path length.** The path length is the number of hops along a path between two nodes; it is the number of circuits a packet must travel across to reach its destination. The path length never exceeds a maximum number of hops, a value that the system manager sets relative to the size and configuration of each network. For an area network, the network manager should determine the maximum number of hops permitted within an area and between areas.
- **Cost.** The cost is an integer value assigned to a circuit between two adjacent nodes. It is usually proportioned to transmission delay. Each circuit has a separate cost. In terms of the routing algorithm, packets are routed on paths with the least cost. Nodes on either end of a circuit can assign different costs to the same circuit.
- **Path cost.** The path cost is the sum of the circuit costs along a path between two nodes. The path cost never exceeds a maximum cost value the network manager specifies for the network. For an area network, the network manager sets the maximum cost for a path within an area, and for a path between areas.
- **Reachable node.** A reachable node is a destination node to which the Routing layer on the local node has a usable path; that is, the path does not exceed the values for maximum cost or hops between nodes specified in the executor database. For an area network, a reachable area is one

to which the path does not exceed the values for maximum cost or hops between areas set in the executor database.

- **Maximum visits.** The maximum number of nodes through which a packet can be routed before arriving at the destination node is referred to as the maximum number of visits the packet can make. If a packet exceeds the maximum number of visits, the packet is dropped.

When configuring a network, the network manager establishes the routing parameters for circuit cost control and route-through control. These parameters allow you to control the path that data is likely to take when being transmitted through the network, and also to minimize congestion at particular nodes in the network. For most networks, the default values for these parameters are reasonable.

The network manager must assign a circuit cost to every circuit that connects the local node with adjacent remote nodes. These costs serve as values that DECnet software uses to determine the path over which data is transmitted. When the node is up and running, you can dynamically change the cost of a circuit to a higher or lower value. Altering circuit costs can change packet routing paths and thereby affect the use and availability of network circuits and resources.

Along with defining circuit costs, you should also consider the path lengths and total path cost for routing packets over the network. For routing purposes, DECnet software identifies the least costly path to each destination in the network. As network manager, you are responsible for defining both the maximum cost of all circuits and the maximum hops that a packet can take when routed to the destination node. If you are configuring an area network, you should define the maximum cost and hops for a path between nodes within your own area, and the maximum cost and hops for a path between level 2 routers in the whole network.

If multiple paths to a destination node have the same path cost, the Routing layer software, by default, splits packet loads for routing on several paths, rather than on only one. This method of **equal cost path splitting** improves network throughput. You can define the maximum number of equal cost paths to be used for routing when a packet load is to be split.

Because equal cost path splitting implies that data packets are sent to the destination node over different paths, the packets may be received out of order by the destination node. The Network Services Protocol (NSP) maintains a cache of out-of-order packets so that they can be reassembled in order. This mechanism is called **out-of-order packet caching**, and is supported by DECnet–VAX Version 4.6 and higher. When packet loads are split and routed to a node that does not support out-of-order packet caching, the destination node is unable to reassemble any packets received out of order. Any packets received out of order by a node that does not support out-of-order packet caching need to be retransmitted. This need for retransmission hinders network performance. You can compensate for a node that does not support out-of-order packet caching by setting the appropriate value for the executor parameter PATH SPLIT POLICY for that node.

The Routing layer in each node of the network uses congestion-control algorithms to maintain an efficient level of routing throughput. In addition, as network manager, you can maintain indirect control over routing throughput by defining the maximum visits a packet can make before being received by the destination node. Packets that exceed this limit are discarded. This control prevents packets from looping endlessly through the network.

2.4.7 Routing Messages

Adjacent routing nodes exchange routing update messages. A routing update message is a packet that contains information about the cost and hops for each node in the network. In an area network, a level 1 router sends routing update messages about all nodes within its own area to adjacent routers in the area. Level 2 routers send routing update messages containing cost and hop information about all areas to adjacent level 2 routers in the network.

Whenever this routing information changes (for instance, when a circuit goes down), new routing messages are sent automatically. For example, if someone were to change the state of a circuit, rendering a remote node unreachable, this change would be reflected automatically in the routing update messages exchanged by the routing nodes.

2.4.7.1 Segmented Routing Messages

The number of nodes that Phase IV DECnet can support in a single-area network is increased to a maximum of 1023 from the limit of 256 for Phase III DECnet. This increase is due to changes in the routing update messages. In Phase III, a legal network was restricted in size to the number of nodes for which cost and hop information could be fit into a single routing update message. Furthermore, Phase III routers had to send complete updates containing information about all nodes, whether or not their reachability had changed. Phase IV allows segmented routing messages to be sent, that is, messages that contain only the information that has been changed. Phase IV also permits routing updates to be sent in multiple messages. Therefore, the size of the routing messages and the number of buffers required to receive them are reduced.

2.4.7.2 Timing of Routing Message Transmissions

The network manager can set a timer for transmission of routing messages, controlling the intervals at which nonconfiguration change routing updates are transmitted. The routing timer controls the frequency of transmission of these messages on non-Ethernet circuits. The broadcast routing timer controls their frequency for Ethernet circuits. Expiration of the broadcast routing timer causes the local node to send a multicast routing configuration message to all routers on the Ethernet.

2.5 Logical Links

DECnet uses a mechanism called a logical link to allow communication between processes running on either the same node or on separate nodes in the network. A logical link carries a stream (consisting of regular data and interrupt data) of full-duplex traffic between two user-level processes. Each logical link is a temporary data path that exists until one of the two processes terminates the connection.

The system manager can control various aspects of logical link operation on the local node. The system manager can do the following:

- Define the maximum number of logical links that can be active at the local node. If your node can also use an alias node address (which is common to some or all nodes in a VAXcluster), you can specify the maximum number of logical links that can use the alias for incoming and outgoing connections. Note that the upper limit on the number of logical links that your node can originate using the individual node address is reduced if your node also uses an alias.

- Specify the number of packets that can be transmitted on a logical link before an acknowledgment is received (the pipeline quota).
- Selectively disconnect active links on the local node while the network is running and verify that the links have been disconnected, by displaying information about the status of the links.

Logical link activity related to NSP is controlled by certain parameters that regulate the duration of NSP connect sequences and inactivity intervals, and the frequency with which NSP retransmits messages. The timers that affect this activity include the following:

- The incoming timer, which protects the local node against the overhead caused by a local process that does not respond to an inbound connection request within a specified interval
- The outgoing timer, which protects the local node against the overhead caused by a connection request to a remote node that does not complete within a specified interval
- The inactivity timer, which protects the user against a link that may be permanently unusable, by setting the frequency with which DECnet tests an inactive link

You should normally use default values for the parameters that regulate the frequency of NSP message retransmission at the local node, unless you need to change the operating characteristics of a particular logical link. The retransmit time is affected by the estimated delay in round-trip transmission between the local node and the node with which it is communicating. You use the delay weight and delay factor parameters to calculate new values for this estimated delay. The retransmit factor parameter governs the number of times NSP tries to retransmit on a logical link.

2.6 Objects

Objects provide known general-purpose network services. An object is identified by object type, which is a discrete numeric identifier for either a user task or a DECnet program such as the Network Management Listener (NML) or the File Access Listener (FAL). The DECnet network software uses object type numbers to enable logical link communication using NSP. The system manager is responsible for supplying information for those objects, both user-defined and network objects, that can be used over the network.

For VAX PSI network operations, you are responsible for identifying objects by name, and establishing command procedures to be initiated when incoming X.25 calls to the objects arrive.

2.6.1 DECnet–VAX Objects

When setting up the network, you must supply information for two general kinds of DECnet–VAX object:

- Objects with a 0 object type. These objects are usually user-defined images for special-purpose applications. They are named when a user requests a connection. Objects in this category are defined in the DECnet–VAX configuration database as TASK (see Section 3.9.1). The object type number for all of these objects is 0.
- Nonzero objects. Nonzero objects are known objects that provide specific network services such as FAL (used for file access) or NML (used for network management). They may also be user-defined tasks; these objects should be for user-supplied known services. Object type numbers for all nonzero objects range from 1 to 255. The number serves as a standard addressing mechanism across a heterogeneous network. For a complete list of network objects, refer to the *VMS Network Control Program Manual.*

The following DIGITAL-supplied objects are defined inside NETACP, by default, in the configuration database. Note that MAIL and PHONE are specific to the VMS operating system.

- File Access Listener (FAL)—an image that provides authorized access to the file system of a DECnet node on behalf of processes executing on any node in the network. FAL communicates with the initiating node by means of the Data Access Protocol (DAP).
- Network Management Listener (NML)—an image that provides services such as gathering and reporting information about network status, zeroing line and node **counters**, and loading a standalone system image to a remote node.
- Event logger (EVL)—an image that logs significant events (locally or remotely) for a given network component.
- Loopback mirror (MIRROR)—an image used for particular forms of loopback testing.
- DECnet Test Receiver (DTR)—a DECnet test program used with the DECnet Test Sender (DTS) to test logical links. The DTS/DTR Utility is described in the *VMS DECnet Test Sender/DECnet Test Receiver Utility Manual.*
- MAIL—an image that provides personal mail service for VMS nodes.
- PHONE—an image that allows you to have online "conversations" with users on the VMS operating system.
- Host loader (HLD)—an image that provides downline task-loading support for RSX–11S tasks.

For every object that can be started by an inbound connection request, you must supply a command procedure, unless either of the following conditions exist:

- The object is one of the following DIGITAL-supplied command procedures: FAL, HLD, NML, EVL, DTR, MAIL, PHONE, MIRROR

- The object is defined as an image, through specification of objectname.EXE as the object file name.

Chapter 3 provides rules for establishing and identifying command files for objects.

You can also specify privileges a user must have in order to connect to the object, and provide default access control information to be used for inbound connections to the object when no access control is specified by the remote node. Additionally, you can assign default proxy login access controls for the object. Refer to Section 2.10 for a discussion of access control information used for logical link connections and a description of proxy login access control.

2.6.2 Objects Using the Cluster Alias Node Identifier

If your node is in a VAXcluster that is using an alias node identifier, you have the option of specifying how the cluster alias node address is to be used in relation to incoming and outgoing connections for specific network objects. By default, all objects except PHONE are able to receive connect requests directed to the alias node identifier. For outgoing connections, the default is that only the MAIL object is associated with the alias node address. If you send mail from a cluster node that uses the alias, the sender's address on the mail message is the alias node identifier.

You should not specify the alias node address for objects that require multiple incoming links, because an incoming link identified by the alias node address may be assigned to any of the nodes participating in the cluster alias node address. For example, PHONE should not use the alias node address, because it requires all incoming links to be directed to the same node in the cluster. Nontransparent tasks that have a mailbox and can receive multiple inbound connection requests should not be accessed using the alias node address (see Chapter 8). Also, objects whose resources are not available clusterwide should not be allowed to receive incoming connect requests addressed to the alias node address.

2.6.3 Creating DECnet–VAX Network Server Processes

On the VMS operating system, all DECnet objects run as processes. Unless a currently running process has declared itself to be a numbered network object or a named network object (with number 0), NETACP must invoke a process to receive the connect request. When the logical link request comes in, a standard procedure called NETSERVER.COM is run, which in turn causes NETSERVER.EXE to be executed. This program works in concert with NETACP to invoke the proper program for the requested object. Then, when the logical link is disconnected, the "object" program (such as FAL) terminates, but the process is not deleted. Instead, control returns to the NETSERVER.EXE program, which asks NETACP for another incoming logical link request to process. This cycle continues until NETSERVER is deleted after a specified time limit. The default is 5 minutes. To use a different default time limit, specify the SYSTEM logical name NETSERVER$TIMEOUT, using an equivalence string in the standard VMS "delta time" format:

```
dddd hh:mm:ss.cc
```

The effect of NETSERVER is to reuse network server processes for more than one logical link request, eliminating the overhead of process creation for an often-used node. NETACP reuses a NETSERVER process only if the access control on the connect request matches that used to start the process originally.

When NETACP creates a process to receive the connect request, the process runs like a batch job. The sequence is as follows:

1. The process is logged in according to information found in the UAF. The key to this file is the user name, which is part of the access control information. The process is successfully logged in only if the password from the access control string matches the password in the UAF record. (Refer to Section 2.10 for a discussion of DECnet access control.)
2. DECnet–VAX automatically creates a log file in SYS$LOGIN:NETSERVER.LOG. Unlike the log file for a batch job, this log file is neither printed nor deleted. The log file is helpful for debugging your own network tasks. If NETSERVER.LOG cannot be created for any reason, the network job continues running but does not produce any log file.
3. The login command procedure indicated in the UAF for the process is executed.
4. The process runs a command file to start the image that implements the DECnet object. The rules for locating this command file differ depending on whether the object has the number 0.

Because NETSERVER.LOG files are not required for network server processes, you may explicitly inhibit all log files in your default nonprivileged DECnet account by setting the default directory for the account to a nonexistent directory. The effect of this action is to suppress all log files, while allowing network jobs to be run.

2.6.4 Potential Causes of Network Process Failures

If a logical link fails and the status information displayed is "network partner exited," this message indicates a problem in the remote network server process. To determine the details of the failure, consult the NETSERVER.LOG file at the remote node. Common reasons for failure are as follows:

- Inability to log in because of failure to access the system login procedure, or the account login procedure or any files that it accesses.
- Protection set on network procedures and images in SYS$SYSTEM, such as NETSERVER.COM or NETSERVER.EXE.
- Attempted execution in your LOGIN.COM file of an interactive command that does not apply to network/batch jobs (for example, a SET TERMINAL/VT100 or SET TERMINAL/INQUIRE command). These commands should not be specified in your LOGIN.COM file unless they are preceded by IF F$MODE() .EQS. "INTERACTIVE".

For example, in your LOGIN.COM file, use the following to prevent a logical link failure:

```
$ IF F$MODE() .EQS. "INTERACTIVE" THEN -
  SET TERMINAL/VT100
```

Any failure to create NETSERVER.LOG causes a network job to continue running, but without a log file.

2.6.5 VAX PSI Objects

The object component of VAX PSI contains records that identify the object, specify a command procedure that is initiated when the incoming call arrives, and specify account information for the incoming call.

You must identify each VAX PSI object by a unique name. For each object, you must create a command procedure for starting the object that is to be executed each time an incoming X.25 call to the object is received. Rules for establishing and identifying the command procedures are given in Chapter 3. For each object, you must also supply account information (consisting of a user name, password, and, optionally, an account name) to be used by incoming X.25 calls from remote DTEs.

2.7 X.25 and X.29 Server Modules

To handle calls coming in over a PSDN from remote DTEs and terminals, you configure the X.25 and X.29 server modules, referred to as the X.25 and X.29 call handlers, as required. The X.25 server module handles incoming calls that originated at a remote DTE; the X.29 server module handles incoming calls that originated at a remote terminal. Your local DECnet–VAX node can receive X.25 and X.29 calls if it is configured as any of the following:

- A DTE connected directly to a PSDN (to be a DTE, a DECnet–VAX node must have VAX PSI software installed)
- A multihost connector node that forwards calls between a PSDN and host node (to serve as a connector node, a DECnet–VAX node must be configured with VAX PSI software in multihost mode)
- A host node that uses a connector node to send and receive X.25 and X.29 calls (to be a host node, a DECnet–VAX node must be configured with VAX PSI Access software)

2.7.1 Destination of Calls from a Remote DTE

The configuration database for the server modules defines the processes that are the destinations for calls, so that incoming calls from a PSDN can be directed to the appropriate destination. If your node is serving as a DTE connected directly to a PSDN or indirectly as a host node using an X.25 connector node, the destination is on the local node. If your node is serving as an X.25 connector node, the destination may be on one of the host nodes using the connector node.

The server database specifies the maximum number of circuits the server module may have, that is, the maximum number of incoming and outgoing calls that all destinations can handle.

When establishing the server configuration database, you must identify each destination by a unique alphanumeric name. You must also name the object activated when a particular destination accepts an incoming call, and assign priorities to all destinations that could handle the same incoming call.

Optionally, you can restrict the incoming calls a destination will handle to any combination of the following:

- Calls to specified local DTE subaddresses
- Calls from specified user groups (BCUGs and CUGs)
- Calls from specified remote DTEs
- Calls containing user data that matches a specified call value after a mask is applied
- Calls that have been redirected from another DTE
- Calls received from a particular network
- Calls containing particular values in the called address extension after a mask is applied

If your local node is serving as an X.25 connector node, you must identify in the X.25 server database the host node on which each destination is located.

Chapter 3 describes how to use NCP commands to specify call-handling parameters in the configuration database.

2.7.2 Handling Incoming Calls at the Local DTE

This section describes the process of handling incoming calls at the local DTE as it relates to network management. The *VAX P.S.I. X.25 Programmer's Guide* describes the handling of incoming calls as it relates to programming PSI network tasks.

Whenever a remote DTE attempts to communicate with your local DTE (that is, attempts to set up a virtual circuit), the remote DTE, the PSDN, and the local DTE provide information that identifies the user process that is the destination of the call. Remote DTE information passed with the call is optional; such information may include the remote DTE address, a local DTE subaddress, a closed user group (CUG) or bilateral closed user group (BCUG) name, and so on. VAX PSI at the local DTE uses this information, along with destination information defined in the configuration database at the local DTE for the server module and objects, to determine how to handle the incoming call.

When an incoming call is received, VAX PSI constructs a network connect block (NCB) using the remote DTE address and other information that may be specified. VAX PSI then attempts to match the information in these fields with the information specified for destinations in the configuration database.

If only one match is made, VAX PSI associates the incoming call with the object specified in the configuration database for this destination. If more than one match is made, VAX PSI chooses the destination with the highest priority. Then it associates the incoming call with the object specified for this destination (see Section 2.6.5).

The object names the task that is to run as a result of the incoming call. (A command procedure associated with the object is activated when the incoming call arrives; this user-written command procedure may activate a user program or an image.) Section 2.6 discusses objects and object parameters specified in the configuration database.

VAX PSI rejects the incoming call if no match is made. To avoid this rejection, you can specify a last-chance destination. A last-chance destination is a destination with an associated object that handles all incoming calls for which a match cannot be found. The simplest last-chance destination is one that specifies the complete range of local DTE subaddresses, handles calls from all DTEs and all user groups, and ignores any incoming call-handling information in the user data field. It specifies no subaddress, no CUGs, no remote DTEs, and no user data. The last-chance destination must have the lowest priority of all the destinations.

2.8 X.25 Access Module

The X.25 access module provides a means for user processes on VMS host nodes to access remote nodes or terminals connected to a PSDN through a connector node. You must configure the host node with VAX PSI Access software. You can configure the connector node with VAX PSI software in multihost mode; the connector node may be an Ethernet communications server, for example, the X25router.

The PSI X.25 access module identifies the connector node to which the local node is to be connected, the network the connector node can access, and, optionally, access control information. The DECnet–VAX host system with VAX PSI Access uses a DECnet link to connect to the connector node. VAX PSI Access software uses the link to transmit X.25/X.29 messages between the host and the connector node.

2.9 Logging

The network software logs significant events that occur during network operation. An event is defined as a network or system-specific occurrence for which the logging component maintains a record. Following is a partial list of significant events:

- Circuit and node counter activity
- Changes in circuit, line, and node states
- Service requests (when a circuit or line is put in an automatic service state)
- Passive loopback (when the executor is looping back test messages)
- Routing performance and error counters (circuit, line, node, and data packet transmission)
- Data transmission performance and error counters (when errors in data transmission occur)
- Lost event reporting (when some number of events are not logged)

This information can be useful for maintaining the network because it can be recorded continuously by the event logger. The system manager is responsible for controlling certain aspects of event logging. In particular, you can control source-related parameters (actual events to be logged, the source for these events, and the location at which these events will be logged) and sink-related parameters (the name of the logging component at the local node and its operational state).

2.9 Logging

For the most part, events are logged for the various DNA layers and for system-specific resources. Events are defined by class and type, in the format class.type. The class of an event identifies the layer or resource to which the event applies, and the type is the particular form of event within the class. For example, event 4.3 indicates oversized packet loss (type 3) for the Routing layer (class 4). Event classes and types are summarized in the *VMS Network Control Program Manual.*

The logging component is the device or process that records logging events. There are three logging components:

- A **logging console**, which is generally a terminal or file that records events in user-readable form. If you do not specify a logging console name, the operator console (OPA0) is used.
- A **logging file**, in which events are recorded in binary format. You can obtain detailed information about the format from the *DNA Phase IV Network Management Functional Specification.* There is no default logging file name.
- A **logging monitor**, which is a program supplied by the system or user that receives and processes events. If you specify a logging monitor, events formatted in user-readable form are sent to the Operator Communication (OPCOM) facility; all network operator terminals (terminals enabled through specification of the DCL command REPLY/ENABLE=NETWORK) display these events. Also, if you specify a logging monitor name, events encoded in binary format are sent to the DECnet object specified by that name. You can obtain detailed information about the format from the *DNA Phase IV Network Management Functional Specification.*

You can use both the logging console and the logging monitor to display events at the operator console; however, the inherent flexibility of OPCOM and its ability to display messages at terminals being used for timesharing may make the logging monitor a more suitable choice for many sites.

The source of an event can be an area, node, module, circuit, or line. Events can be logged at either the local node or a remote node; this node is called the **sink node**.

At the local node, you can control the operational state of the logging sink. You must turn logging on before events can be logged to the sink, and off before the logging parameters for the sink can be cleared from the database. You specify the hold state to queue events for a specific logging sink.

2.10 Network Access Control

DECnet–VAX regulates access to the network on various levels, including the following:

- Routing initialization passwords for links connecting the local node to remote nodes
- System-level access control for inbound logical link connections that result in a process being created
- Node-level access control for inbound and outbound logical links
- Proxy login access control for individual accounts

The following sections describe these levels of control as they relate to DECnet–VAX software operation, from the perspective of the system manager's need to establish control parameters through NCP. Chapter 3 describes how to use specific NCP commands to accomplish access control.

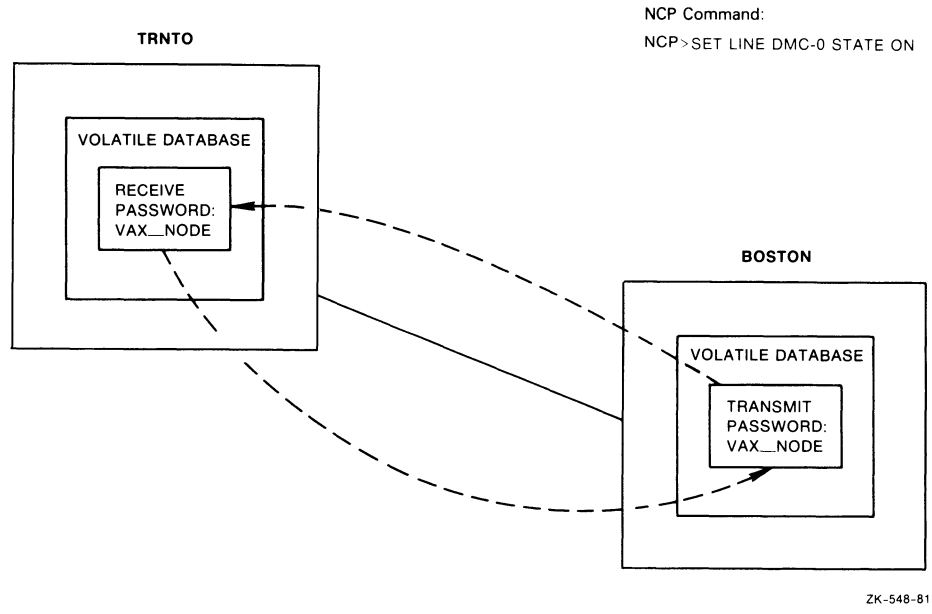
2.10.1 Routing Initialization Passwords

Whenever you turn on a circuit, your local node attempts to initialize with the DECnet software at the remote node connection for that circuit. As part of this initialization process, the remote node may require a password to complete the operation. The system manager can specify passwords when setting up the configuration database.

In a Phase IV network, passwords are required when a Phase III node is initialized (see Section A.5.2.2), but are optional when a Phase IV node is initialized. Generally, passwords are used in initializing Phase IV nodes only when a system has dialup telephone lines used by the network. When a dialup node seeks a dynamic connection over a terminal line, the dialup node must supply a password, but the node receiving the login request does not send a password to the dialup node.

Figure 2–4 illustrates a routing initialization sequence for the network example. When the circuit is turned on, nodes BOSTON and TRNTO initialize. On the local node, BOSTON, DECnet–VAX software retrieves the transmit password for remote node TRNTO and sends it to TRNTO upon request. On node TRNTO, DECnet–VAX verifies this password with the receive password specified for remote node BOSTON in its configuration database. After the passwords are verified, the link is operational; that is, the circuit state makes the transition from ON-STARTING to ON.

Figure 2–4 Routing Initialization Passwords



DECnet–VAX always solicits a receive password. However, if verification on the circuit is disabled, or if no receive password is specified in the database for the adjacent node, DECnet–VAX accepts anything the adjacent node may send. The adjacent node is still required to send the verification message.

2.10.2 System-Level Access Control

DECnet–VAX provides system-level access control over logical link connections. The network user on the initiating node may explicitly supply an access control string to control which account is used on the remote node. If, however, the initiating node does not supply explicit access control information, DECnet optionally provides default access control when sending the request to the remote node. It also optionally provides default access control for incoming logical links if the initiating node has not supplied access control information.

2.10.2.1 Setting Access Control Information for Outbound Connects

The system manager can specify default access control information for outbound connections. This enables the local node to send outbound logical link requests with default access control information when that information is not explicitly provided. The remote node stores the access control information in its configuration database. The default access control information can include privileged and nonprivileged names and passwords to be used in connecting to a particular remote node.

The system manager at a node can specify a list of privileges required for connection to a particular object, such as NML. When the local node requests connection to an object for which privileges have been specified, it sends the default privileged access control string to the remote node. If the system manager does not specify privileges for an object, such as FAL, the object is accessible to all users. When the local node requests connection to this object, it sends the nonprivileged access control string.

2.10.2.2 Sources of Access Control Information for Logical Link Connections

Whenever a local DECnet node attempts to connect to a remote DECnet–VAX node by means of a logical link, system-level access control information is sent to the LOGINOUT image running in the context of a process on the remote node. Access control information can come from a number of sources:

- The network user on the local node may explicitly supply an access control string. If this is the case, the remote node uses the access control information.
- If the access control string is not explicitly supplied, the local node checks its object database against the privileges of the initiating process. If the object does not require privileges other than TMPMBX and NETMBX, the local node sends the default nonprivileged access control string from its node database to the remote node.
- If the object requires privileges beyond TMPMBX and NETMBX, and the user process has the required privileges, the local node sends the default privileged access control string from its node database to the remote node.
- If no access control string is supplied, the local node checks to see if proxy access is enabled for the remote node. If so, LOGINOUT at the remote node checks the NETPROXY.DAT file to determine whether a user should be logged in to a designated account rather than the nonprivileged account. (Proxy login access control is described in Section 2.10.5.)

- If none of these cases are valid, the local node sends a "no" access control string.
- When the remote node sees that no access control has been specified, it checks its object database. If the object database contains a default inbound access control string, the remote node uses that string.
- If there is no default access control information in its object database, the remote node checks its executor node database for nonprivileged account information for itself. If the information is there, the remote node uses the nonprivileged access control string.

Finally, if none of these sources supply the information, the connection fails.

Note: In DECnet-VAX usage, nonprivileged means NETMBX and TMPMBX privileges only. NETMBX is the minimal requirement for any network activity. Privileged means any privileges in addition to NETMBX or TMPMBX.

Figure 2–5 illustrates the local node's access control options for inbound connection requests.

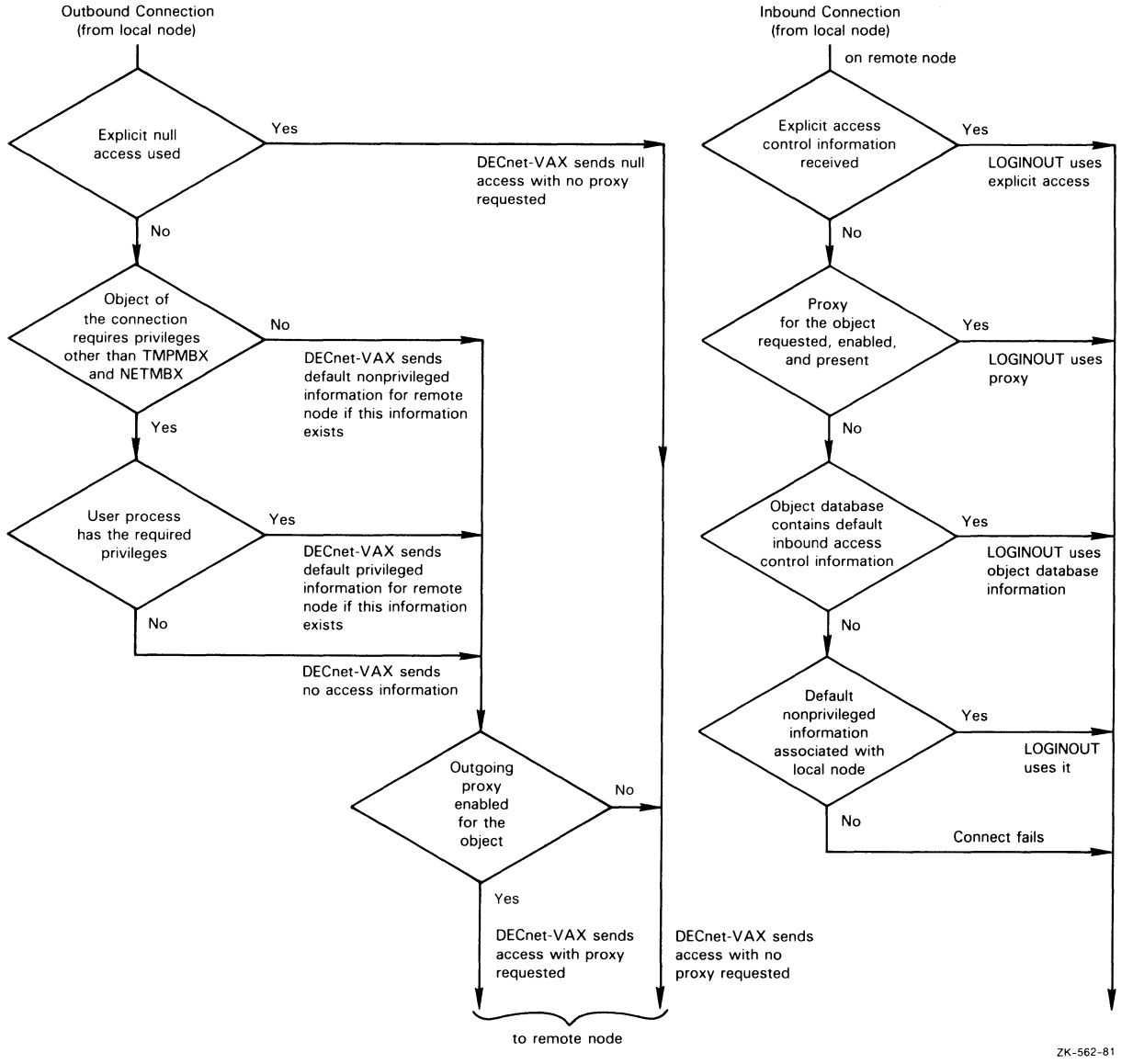
Regardless of the source, the remote node uses this access control information to determine whether a logical link can be established. The way this validation process works is important for both the system manager and network users. This section discusses access control in terms of network management. Chapter 8 discusses access control as it relates to user-level operations such as remote file access and task-to-task communication.

Access control information is not used where the connection is to a program that has declared a name or object number and has started independently of DECnet.

Access control information allows users on remote nodes to gain access to resources on the local node. The system manager must establish access control information in both the configuration database (for objects) and the UAF (for default network accounts) on the local node. Chapter 1 briefly describes the necessity for these default accounts. Chapter 5 explains how to create a default nonprivileged DECnet account.

Whenever NETACP on the local node receives an inbound logical link connection request, it creates a process and starts the LOGINOUT image, which verifies the user's access rights by checking the UAF. When the VMS operating system starts a user process as a result of an inbound connection request, the privileges with which that process runs are determined by the UAF record associated with the access control information passed in the connection message. This function is almost identical to the one that occurs whenever a local user starts a batch job; the difference is that the resulting LOG file is neither printed nor deleted. Section 2.6 discusses this process in detail.

Figure 2–5 Access Control for Inbound Connections



2.10.2.3 Network Security and Passwords

You can maintain password security in a network environment by protecting the network configuration files from unauthorized access. The most convenient way to do this is to require SYSPRV to access these files. NML must have access to network configuration files. NML on the remote node accesses these files when it sees the NCP commands that access the permanent database (DEFINE, PURGE, LIST, and SET "component" ALL).

After you set access control (as described in the previous section), users must have the privileges necessary to perform the following operations:

- To modify the volatile database, NML users must have OPER privilege.

- To specify the permanent database (using the DEFINE command) or to update the volatile database with all parameters from the permanent database (using the SET "component" ALL command), users must have OPER privilege and WRITE access to all permanent database files.
- To remove specific parameters from the permanent database (using the PURGE command) or to reset or remove all parameters from the volatile database (using the CLEAR "component" ALL command), users must have OPER privilege and write access to all permanent database files. To clear counters, users must have OPER privilege.
- To start the network, users must have ACNT, CMKRNL, SYSNAM, and DETACH privileges.

To make these safeguards operational, you should avoid assigning privileges beyond those normally used. In particular, you should not give the default privileged account SYSPRV. These default accounts should be in their own group to avoid extending group access to other directories on the local node. You can protect sensitive files and directories against world access by requiring explicit access control to reach them.

2.10.2.4 Inbound Default Access Control for Objects

Another form of access control specific to network objects is default account information used by inbound connects from remote nodes that send no access control information. Because no access control information is supplied, the default information you specify for the object is used to allow the logical link connection to be made. One example of this is downline task loading. When SLD connects to HLD on the host node, default access control information specified for the HLD object is used. Refer to Chapter 4 for more information about downline task loading.

2.10.3 Access Control for Remote Command Execution

If you request an NCP command to be executed at a remote node, you can supply an explicit access control string or default to access control information in the configuration database. To supply an explicit access control string, you use either the standard VMS node specification *node"user password account"*:: or specify this access control information as parameters in the NCP command to be executed at a remote node.

2.10.4 Node-Level Access Control

The system manager can regulate two forms of node-level access control for incoming and outgoing logical links. One form involves specifying the ACCESS parameter for a particular node in your volatile database, and the other involves specifying the DEFAULT ACCESS parameter in your executor database.

When an incoming or outgoing logical link connection is attempted, the executor node first checks its volatile database for the ACCESS entry for the target node. If the entry exists, the executor uses it.

Because it may not be feasible to include an ACCESS entry for every node in a large network, DECnet–VAX provides the DEFAULT ACCESS alternative. If the logical link connection is attempted and there is no ACCESS entry for the remote node in the volatile database, the executor uses the DEFAULT ACCESS parameter value.

Both commands accept the same set of parameter values, which are as follows:

INCOMING	Allows logical link connections from the remote node, but does not allow the local node to initiate connections to the remote node.
OUTGOING	Allows the local node to initiate connections to the remote node, but does not allow connections from the remote node.
BOTH	Allows incoming and outgoing logical link connections. This is the default.
NONE	Does not allow incoming or outgoing logical link connections to this node.

If you specify no entry for the ACCESS or DEFAULT ACCESS parameter, the DEFAULT ACCESS parameter defaults to BOTH.

Only those users with OPER privilege can bypass this access protection.

For each node, you can configure the privileged and nonprivileged accounts and passwords that constitute default access control information. This default access control information should match the system-level access control information established for the node (see Section 2.10.2).

Another form of access control at the node level is the node checking that is performed before a system can dial in and form a dynamic asynchronous connection over a terminal line. For a description of security measures for dynamic asynchronous connections, see Section 2.10.6.

2.10.5 Proxy Login Access Control

Proxy login enables a user logged in at a remote node to be logged in automatically to a specific account at the local node, without having to supply any access control information. Note that proxy login is not the same as interactive login. Proxy login means that specific network access operations can be executed. By contrast, interactive login requires a user to supply a username and password before the user can perform any interactive operations. In order to establish proxy login to an account on the local node (without specifying any access control information), the remote user must have a default proxy account on the local node that maps to a local user account. The remote user assumes the same file access rights as the local account and also receives the default privileges of the local account. You can use the proxy login capability to increase security, because it minimizes the need to specify explicit access control strings in node specifications passed over the network or stored in command procedures.

Note that network objects can also be assigned proxy login access.

The following sections summarize the procedures for establishing proxy accounts and for establishing proxy access to network objects.

2.10.5.1 Proxy Accounts

Proxy accounts permit users on remote nodes to obtain access privileges on other nodes without having private accounts on those nodes. The remote user can enter commands to access data that is accessible by one or more local accounts to which that remote user has proxy access.

A system manager can control the use of proxy accounts at the local node, by using the Authorize Utility to create and modify the permanent proxy database, NETPROXY.DAT. In NETPROXY.DAT, each database entry maps a single remote user to one or more local accounts. The remote user is identified by either a node name and a user name, or by a node name and a remote UIC (the User Identification Code used by the Authorize Utility). The following examples show how remote users may be identified in the proxy database:

```
LARK::KELLEY
RSTS::[23,55]
```

In the first example, the remote user is identified by the node name LARK and user name KELLEY. The second example specifies that a UIC is to be used instead of a user name to identify the user as member 55 in group 23.

In the permanent proxy database, each remote user may be mapped to one default proxy account and up to 15 additional proxy accounts on the local node. With a default proxy account, the remote user does not need to specify a user name or password when requesting proxy login. With a nondefault proxy account, the remote user must include a user name only.

For a summary description of proxy accounts and how to create them, see the *Guide to VMS System Security*. The Authorize Utility is described in the *VMS Authorize Utility Manual*.

2.10.5.2 Controlling Proxy Login Access for Individual Accounts

The permanent proxy database resides in NETPROXY.DAT. All management and maintenance of this database is handled through the Authorize Utility. NETPROXY.DAT is updated automatically any time you use the Authorize Utility to make any changes to proxy logins. When DECnet is started up, the information in NETPROXY.DAT is used to construct a volatile database in the NETACP process. This volatile database is consulted by NETACP when incoming proxy login requests are received at the local node.

When the local node receives a request for initiation of an inbound connection, and if no access control string is supplied and the remote node is enabled for outgoing proxy login access, the local system checks to see if the object has incoming proxy enabled. If proxy access is enabled, NETACP checks its volatile database to determine whether the user should be allowed to log into a designated account.

By default, both incoming and outgoing proxy login access are enabled at the local (executor) node. Consequently, incoming and outgoing proxy login access is permitted with all remote nodes. These default values are established by DECnet–VAX to permit proxy logins to be initiated by the local node or by the remote node. These default values are the recommended settings.

However, you can restrict the use of proxy logins by specifying the NCP executor parameters INCOMING PROXY and OUTGOING PROXY in the volatile database. The possible proxy access options for the local node are as follows:

INCOMING PROXY enabled	Allows proxy login access from the remote node to the local node.
INCOMING PROXY disabled	Prevents proxy login access from the remote node to the local node.
OUTGOING PROXY enabled	Allows the local node to initiate proxy login access to the remote node.
OUTGOING PROXY disabled	Prevents outgoing proxy login access connections from the local node.

After the network is started, the NCP command SET KNOWN PROXIES ALL can be used to update the volatile proxy database.

2.10.5.3 Controlling Proxy Login Access for Objects

Just as you can control proxy login access by individual accounts, you can control proxy login access by network objects. You control proxy login access to a specific network object by setting the value of the object parameter PROXY in the configuration database. The database contains defaults for each object. Permitting proxy login access to an object is recommended only if the proxy access serves some useful purpose. For example, by default MAIL is set to prevent incoming proxy login, while FAL is set to allow both incoming and outgoing proxy login.

Note that whatever you declare for the object proxy database takes precedence over the values declared in the executor proxy database.

The following four options are available for the PROXY parameter for a network object:

INCOMING	Allows proxy login to the object.
OUTGOING	Allows the object to initiate proxy login.
BOTH	Allows both incoming and outgoing proxy login access. This is the default.
NONE	Does not allow incoming or outgoing proxy login access.

Note that there are advantages to disallowing incoming proxy access to an object (such as MAIL) that does not require it. Whenever possible, incoming connect requests are matched up with compatible existing NETSERVER processes, to avoid the overhead of unnecessary process creation. If the object disallows incoming proxy access, incoming connect requests will use default access control, with a higher probability of being matched with an existing NETSERVER process.

2.10.6 Security for DDCMP Point-to-Point Connections

If a remote node requests a connection over a DDCMP point-to-point circuit, the local node can avoid revealing its routing initialization password, while requiring that the remote node supply its password. This security measure is used to protect the password of the local node when a dialup node initiates an asynchronous connection to the local node.

For example, a user at a system with an asynchronous terminal line (such as a VMS operating system running on a MicroVAX) can dial in to another system (such as a VMS operating system in a VAXcluster) and initiate a dynamic connection. This connection causes the terminal lines to be converted to asynchronous DDCMP communication lines for the duration of the telephone call. To prevent attempts at access by callers at unauthorized nodes, certain checks have been included in the dynamic configuration process. The dialup node must be the type of node (router or end node) expected by the local VMS node. When the dialup node attempts to initialize, it must supply a routing initialization password to the local node, although the local node does not send its password to the dialup node. The line will not be started unless the password can be verified at the local node. This convention preserves the security of the local node in case the dialup node is unauthorized. The line will not be started unless the transmit password sent matches the local receive password. In addition, depending on how the user set up the terminal line, the connection can be configured to end automatically when the telephone is hung up.

Part II Network System Management

3 Managing and Monitoring the Network

This chapter explains how to use network management commands and **parameters** to configure, manage, and monitor network software. The management tools and components available to DECnet–VAX and VAX PSI users fall into 13 broad categories:

- Configuration database
- Network Control Program (NCP)
- Executor node and remote nodes
- X.25 protocol modules
- Circuits
- Lines
- Routing
- Links
- Objects
- X.25 and X.29 server modules
- X.25 access modules
- Logging
- Access control

This chapter provides enough information for you to build a network **configuration database** for your VMS operating system. It also explains how to use most NCP commands at both the local node and remote nodes to modify parameters for the running network. See Chapter 5 for examples that use NCP commands to build databases for various network configurations.

Chapter 2 describes DECnet–VAX and VAX PSI network components and operating concepts. The *VMS Network Control Program Manual* contains reference information about the operation of the Network Control Program (NCP) Utility and the complete syntax of NCP commands.

3.1 The DECnet–VAX Configuration Database

The DECnet–VAX configuration database contains files that provide information about the local node, remote nodes, local physical lines, local circuits, local logging, and local objects. Each DECnet node in the network has a network database that supplies component and parameter information of this kind. To ensure successful node-to-node communication, each node has a configuration database that consists of the following databases:

- A node database with a record for each node, including the local node
- A circuit database with a record for each circuit known to the local node

- A line database with a record for each physical line known to the local node
- A logging database with a record for each sink (logged events are sent to the sinks)

In addition, NETACP provides a default **object database** with a record for each object known to the network, including objects (for example, FAL) that are defined when you bring up the local node.

As system manager, you need to specify the nodes that can communicate with your node, the physical lines that connect the nodes, and the circuits associated with those lines. In some cases, this connection may include more than one line and circuit to the remote node. You also need to establish a variety of operational routing parameters for the local node to ensure proper network operation.

To allow for communication between nodes, NETACP defines several network objects including NML, FAL, and TASK.

To provide network management flexibility, the DECnet–VAX configuration database consists of two distinct databases, one volatile and one permanent. In addition, if VAX PSI is included in the network, a VAX PSI configuration database, consisting of a volatile database and a permanent database, exists at the local DTE (see Section 3.1.3).

3.1.1 The Volatile Database

The volatile copy of the DECnet–VAX configuration database is memory resident; it allows you to control the running network without modifying the permanent database. NCP provides commands for setting, clearing, and showing network component parameters for the **volatile database**. NCP also permits you to copy current information about remote nodes from the node database of another node into your volatile database.

You can change parameters in the volatile database while the system is running; these changes, however, are in effect only until you modify them again or until the network is shut down. NETACP uses parameters specified only in the volatile database.

3.1.2 The Permanent Database

The permanent copy of the DECnet–VAX configuration database provides the initial values for the volatile database. You access the **permanent database** whenever you use the ALL parameter with the SET command, for example, when you bring up the network. In effect, the permanent database allows you to load network parameters into the volatile database when you boot the system. You can also change parameters in the permanent database.

You can use NCP commands to define, purge, and list network component parameters in the permanent database. You can also use NCP to copy current remote node entries into your permanent node database from the database of another node to which you have access.

You can optionally use the NETCONFIG.COM procedure to configure automatically the permanent database for your node (see Chapter 5).

3.1.3 VAX PSI Configuration Database

The VAX PSI configuration database contains files that provide information about the local DTE, local lines, virtual circuits, local modules, and local objects. Each PSI DTE connected to a PSDN has a database that supplies component and parameter information of this kind. For successful DTE-to-DTE communication, each DTE has a minimum configuration database that consists of the following databases:

- A circuit database with a record for each permanent virtual circuit (PVC), if PVCs are in use
- An object database with a record for each object
- A line database with a record for each physical line to a PSDN
- A module database with records for the PSDN(s), the DTE(s), groups, and destinations.

The VAX PSI configuration is also stored in the DECnet–VAX configuration database.

Just as with the DECnet–VAX configuration database, the VAX PSI configuration database consists of both a volatile and a permanent database.

You can use NCP commands to configure the VAX PSI software at any time. The STARTNET.COM command procedure sets the parameters in the volatile database every time you load the VAX PSI software (see Chapter 6). You can change parameters in the configuration database while VAX PSI is running.

3.2 The Network Control Program

The Network Control Program (NCP) is the vehicle for creating and modifying component parameters in the configuration database. In addition to the NCP command interface, DECnet–VAX users can write programs that communicate with NML through the Network Information and Control Exchange (NICE) protocol. For information about this interface, refer to the *DNA Phase IV Network Management Functional Specification.*

Most NCP commands allow you to modify either the volatile or the permanent database. NCP accesses either database, depending on which command verb you use. For example, you enter the following command to access the permanent database to create or modify the address of a remote node:

```
NCP>DEFINE NODE 14 NAME DENVER
```

To change the parameter in the volatile database, you enter the following command:

```
NCP>SET NODE 14 NAME DENVER
```

The following table distinguishes command verbs by function and the database they access.

Function	Volatile	Permanent
Creating/modifying parameters	SET	DEFINE
Clearing parameters	CLEAR	PURGE
Displaying parameters	SHOW	LIST

Because the commands to access the volatile and permanent databases are similar, this section uses volatile database commands in all examples.

When configuring your network, you can use NCP either to build upon previously specified information or to change that information. Thus you do not have to delete all existing parameters and start over. For example, assume that you have identified a remote node address as 5. You can add node parameters for this record in the volatile database by using the SET NODE command. If you want to change the address of this node, you need to specify a new address only in the ADDRESS parameter of the SET NODE command. If you decide later that you want to remove any or all parameters for this node, then you could use the CLEAR NODE command. Commands to remove parameters exist for all network components.

NCP commands operate on network components and their parameters. When issuing an NCP command, you must provide the command verb, the component name, and one or more parameters, qualifiers, or both, as shown in the following example:

```
$ RUN SYS$SYSTEM:NCP

NCP>SET  NODE 11          NAME BOSTON COUNTER TIMER 30
NCP>SET  KNOWN LOGGING    STATE ON
         .
         .
         .

NCP>SET  EXECUTOR         STATE ON
         ~~~~~~~~~~~~     ~~~~~~~~
command  component        parameters
```

Note that components consist of two types: singular (as with NODE BOSTON) and plural (as with KNOWN LOGGING). For example, you can display information about an individual node or all nodes (including the local node) in the network:

```
NCP>SHOW NODE BOSTON COUNTERS
         .
         .
         .

NCP>SHOW KNOWN NODES COUNTERS
         .
         .
         .
```

Most NCP commands support both singular and plural component names.

NCP accepts the asterisk (*) and the percent sign (%) as wildcard characters. You can include these wildcard characters on the NCP command line to represent a group of component names. Using a wildcard character allows you to refer to an NCP component by a general name, rather than by a specific name.

You can use wildcard characters to represent the following component names:

- Node name
- Line name
- Circuit name
- Object name
- Node address
- Events

The asterisk wildcard represents one or more characters, while the percent sign represents a single character.

The following rules define how you can use wildcard characters with component names.

- If the component name is a string, the wildcard character may occur at any location in the string. For example:

  ```
  NCP>LIST NODE ST%R STATUS
  NCP>SHOW OBJECT M* CHARACTERISTICS
  ```

  The first command requests a list of status information for all nodes with four-letter node names beginning with "ST" and ending with "R." The second command requests a listing of characteristics for all objects with names beginning with "M."

- For node addresses, which are represented by the format *area-number.node-number*, only the *node-number* portion of the node address (the numeral on the right side of the period) can contain a wildcard. For example, the following command sets a COUNTER TIMER value of 45 seconds for all nodes in area 4:

  ```
  NCP>SET NODE 4.* COUNTER TIMER 45
  ```

  Specifying a node address such as *.5 is invalid because only the *node-number* can contain a wildcard.

- In a node address, a wildcard character cannot be combined with a numeral to represent a *node-number*. The node addresses 4.* and 4.% contain valid uses of the wildcard characters, but the node addresses 4.%2 and 4.1* are invalid.

- For events, which are represented by the format *class.type*, only the *type* portion of the event (the numeral on the right side of the period) can contain a wildcard. For example, the following command specifies that all class 2 events are to be logged:

  ```
  NCP>SET KNOWN LOGGING EVENTS 2.*
  ```

- Except in the case of events, only component names can contain wildcards. Parameter values cannot contain wildcards. The following command is invalid because the circuit name UNA-* is not the component name in the command. Rather, it is a parameter used to modify the component named BOSTON. Only component names can be represented by wildcard characters.

  ```
  NCP>SET NODE BOSTON SERVICE CIRCUIT UNA-*    !INVALID COMMAND
  ```

The component name EVENT is used as a parameter to the LOGGING commands, and can contain wildcard characters, as long as only the *type* portion of the event number (the numeral to the right of the period) contains the wildcard. For example, the following command clears logging to the logging file for all class 2 events:

```
NCP>CLEAR LOGGING FILE EVENTS 2.*
```

- Unit numbers of circuit and line devices may contain wildcard characters, but device names of circuits and lines cannot contain wildcard characters. Circuit and line devices are typically identified by the format *dev-c*, where *dev* is a mnemonic device name, and *c* is a device unit number. In the following example, the asterisk replaces the unit number in this request for circuit information for all DMC devices:

  ```
  NCP>SHOW CIRCUIT DMC-*
  ```

  However, the *device-name* portion of a circuit or line name cannot contain wildcard characters. Therefore, the following commands are invalid:

  ```
  NCP>SHOW CIRCUIT D* STATUS  !INVALID COMMAND
  NCP>SHOW LINE %NA-0 SUMMARY !INVALID COMMAND
  ```

Note that substituting a wildcard character for an entire component name is equivalent to specifying the command component KNOWN. For example:

```
NCP>SHOW NODE * STATUS
```

This command is equivalent to the following command:

```
NCP>SHOW KNOWN NODES STATUS
```

For a detailed description of NCP operation, the syntax of NCP commands, and examples of NCP command prompting, refer to the *VMS Network Control Program Manual*.

3.3 Node Commands

To establish your VMS operating system as a node in the DECnet network, you must build the node database entries for the DECnet–VAX configuration database. The following sections describe identification of the executor node and remote nodes, and the node parameters required to build an operational network node database. They also discuss how to update your node database by copying current information about remote nodes from another node to which you have access.

3.3.1 Executor Node Commands

NCP allows you to manage the operation and configuration of both your local node and remote nodes in the network. Generally, the NCP commands you enter at your local node are executed on that node. Occasionally, however, you may want to enter commands from the local node to be executed on adjacent or remote nodes. To this end, NCP incorporates the concept of an executor node. The executor node is the node on which NCP functions are actually performed, which in most cases is the local node. To perform NCP functions on remote nodes, NCP supports two commands: SET EXECUTOR NODE and TELL.

Note: For command descriptions in this manual, the executor node on non-Ethernet circuits is assumed to be the local node (BOSTON) unless otherwise specified. On Ethernet circuits, the executor node is usually ROBIN.

3.3.1.1 SET EXECUTOR NODE Command

The SET EXECUTOR NODE command sets the executor to the node at which you want the commands to execute. One use of this feature is to display information about the configuration database of the remote node. Figure 3–1 illustrates this use of a remote executor node.

As shown in Figure 3–1, you set the executor node by entering the following NCP command:

```
NCP>SET EXECUTOR NODE TRNTO
```

NCP executes commands that you enter at your local node, BOSTON, at the remote executor node, TRNTO. The executor node interprets each command with its own network management software, and then performs the NCP function. Each command must be stated as if it were issued to NCP at the executor node. In this example, any information output that results from the execution of a command is displayed at node BOSTON. If the remote node is not running DECnet–VAX software, refer to the appropriate documentation for that node.

To reset the executor to the local node, use the following NCP command:

```
NCP>CLEAR EXECUTOR NODE
```

The executor is always the local node when NCP is activated. Several users at one node can set their executor to different nodes.

When you issue a SET EXECUTOR NODE command, you can either include specific access control information or use the default access control information. The level of privilege allowed at the remote executor node depends on the access control information specified. (Section 3.13 describes the access control format.)

Note: When you clear the executor node, NCP communicates with NML as a shared image in the same process. Hence, clearing the executor node resets the executor's privileges to those of your current process—that is, the process running NCP.

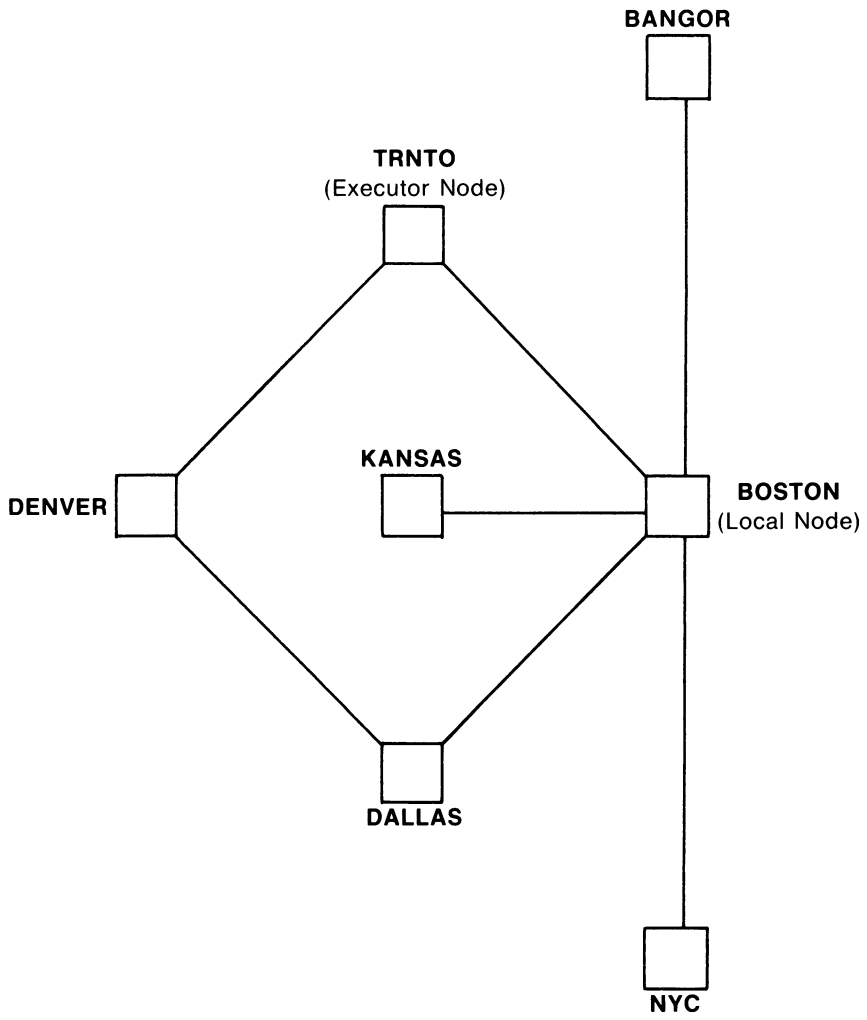
3.3.1.2 TELL Prefix

As an alternative to using the SET EXECUTOR NODE command, you may want to execute only a single command at a remote node or you may want to temporarily override the current executor. In either case you can use the TELL prefix with an NCP command. For example, if you enter the following command at node BOSTON, NCP displays line information for all physical lines connected to node TRNTO:

```
NCP>TELL TRNTO SHOW KNOWN LINES
```

Remote execution in this case applies only to the one command entered with the TELL prefix. Again, you can specify or default to access control information.

Figure 3–1 Remote Command Execution



3.3.2 Node Identification

When configuring the network, you must identify in the executor configuration database the local node and all adjacent nodes connected to it by circuits. Identifying all nodes by name as well as address permits you to reach any node by its name. This section describes node identification and discusses NCP parameters relevant to identifying nodes.

Either the node address or the node name can serve as a node identifier (node-id). The node address is a decimal number assigned to the node in the configuration database. The address must be unique within the network. The node address may include as a prefix the area number, a decimal integer indicating the area in which the node is grouped. In the node address, the area number and node number are separated by a period, in the following format:

area-number.node-number

For example, if node 3 is in area 7, its node address is 7.3. The area number must be unique within the network and the node number must be unique within the area. If you do not specify an area number, the area number of the executor node is used. The default area number for the executor is area 1. In multiple-area networks, you should always specify the area number.

A node name is an optional, unique alphanumeric string that contains up to six characters including at least one alphabetic character. You can use it interchangeably with the node address to identify a node. In the single-area network example in Chapter 1, the node name BOSTON and the node address 1.11 identify the same node.

When defining remote nodes in the volatile database, use the SET NODE command to specify node names and node addresses. The following command associates the node name TRNTO with the node whose address is 1.5:

```
NCP>SET NODE 5 NAME TRNTO
```

To specify a node address for the local node, use the SET EXECUTOR command, as in the following example:

```
NCP>SET EXECUTOR ADDRESS 11
```

Then, use the following command to specify a node name for the local node:

```
NCP>SET NODE 11 NAME BOSTON
```

By entering these commands, you have established a remote node (TRNTO) whose address is 1.5 and the local node (BOSTON) whose address is 1.11. You can then build upon this information to establish parameters for the various nodes.

Before a node can be accessed by name, you must specify a node name to be associated with a node address.

After you set the executor node's address in the volatile database, you cannot change it unless you turn off and restart the network. However, you can change any other node's address at any time. For example:

```
NCP>SET NODE TRNTO ADDRESS 6
NCP>SET NODE TRNTO ADDRESS 8
```

3.3.2.1 MAXIMUM ADDRESS Parameter

The MAXIMUM ADDRESS parameter sets the highest address that the local node will recognize. Setting this parameter allows you to group node addresses in a predefined range, which minimizes the size of internal data structures and control mechanisms for DECnet–VAX software. For example, the following command sets the highest remote node address at 17:

```
NCP>SET EXECUTOR MAXIMUM ADDRESS 17
```

3.3.2.2 Local Node Identification Parameter

In addition to defining a node name and address for the local node, you can also specify a descriptive quoted string of alphanumeric characters. NCP displays this string whenever you enter the SHOW EXECUTOR or LIST EXECUTOR command. Use the IDENTIFICATION parameter with the SET EXECUTOR command to specify this optional information, as follows:

```
NCP>SET EXECUTOR IDENTIFICATION "DECnet-VAX V5.0,  VMS V5.0"
```

This command provides information that NCP displays whenever you use the SHOW EXECUTOR command to display executor node information, as in the following example:

```
NCP>SHOW EXECUTOR CHARACTERISTICS

Node Volatile Characteristics as of 30-DEC-1988 11:27:07

Executor node                 = 1.11 (BOSTON)

Identification                = DECnet-VAX V5.0,  VMS V5.0
Management version            = V4.0.0
Incoming timer                = 45
Outgoing timer                = 45
Incoming Proxy                = Enabled
Outgoing Proxy                = Enabled
NSP version                   = V4.0.0
Maximum links                 = 128
Delay factor                  = 80
Delay weight                  = 5
Inactivity timer              = 60
Retransmit factor             = 10
Routing version               = V2.0.0
Type                          = routing IV
Routing timer                 = 600
Broadcast routing timer       = 40
Maximum address               = 1023
Maximum circuits              = 16
Maximum cost                  = 1022
Maximum hops                  = 15
Maximum visits                = 63
Maximum area                  = 63
Max broadcast nonrouters      = 64
Max broadcast routers         = 32
Maximum path splits           = 1
Area maximum cost             = 1022
Area maximum hops             = 30
Maximum buffers               = 100
Buffer size                   = 576
Default access                = incoming and outgoing
Pipeline quota                = 1200
Alias incoming                = Enabled
Alias maximum links           = 32
Alias Node                    = 1.10 (CLUSTR)
Path split policy             = Normal
```

3.3.2.3 Using and Removing Node Names and Addresses

After you specify a node name and address, you can use them interchangeably whenever you need to specify a node-id. The local DECnet–VAX software translates the node names into node addresses. In the single-area network example, the following NCP commands perform identical functions:

```
NCP>SHOW NODE 5 CHARACTERISTICS
.
.
.
NCP>SHOW NODE TRNTO CHARACTERISTICS
.
.
.
```

To remove a remote node name from the volatile database, use the CLEAR NODE command. The following command removes the association between TRNTO and node 5:

```
NCP>CLEAR NODE 5 NAME TRNTO
```

To remove a remote node address from the volatile database, you must remove all parameters for the node. You can also remove addresses for all known nodes other than the local node, as in the following example:

```
NCP>CLEAR NODE TRNTO ALL
.
.
.
NCP>CLEAR KNOWN NODES ALL
```

After all parameters for a component are removed from the volatile database, the component is no longer recognized by the network.

Note: To change the ADDRESS or BUFFER SIZE parameter for your node, you must first turn off the executor. For information about how to change the local node's operational state, refer to Section 3.3.5.2 and Chapter 6.

3.3.3 Identifying Cluster Nodes

For many network operations, being able to treat nodes within a homogeneous VAXcluster as though they were a single node in a DECnet network is convenient. You can do this by establishing an alias node identifier for the cluster. You can specify the alias node identifier as either a unique node address or a corresponding node name. Any member node can elect to use this special node identifier as an alias while retaining its own unique node identification. Use of the cluster alias node identifier is optional.

The management of a cluster alias node involves three primary decisions:

1. Will an individual node participate in the use of a cluster alias node identifier?
2. If a node participates, does it want to receive inbound connect requests targeted to the cluster alias address?

3 For any object defined on a participating node, should the object's logical links appear to have originated from the cluster alias node and should the object be able to receive incoming connect requests that are directed to the cluster alias address?

To establish an alias node identifier for a local node, use the SET EXECUTOR or DEFINE EXECUTOR command with the ALIAS NODE parameter, described in Section 3.3.3.1. To enable incoming requests to the cluster alias node address, use the ALIAS INCOMING parameter of the SET EXECUTOR or DEFINE EXECUTOR command, as described in Section 3.3.3.2.

The SET OBJECT command allows you to associate specific objects with the cluster alias node identifier, by means of the ALIAS OUTGOING parameter. You can also use the ALIAS INCOMING parameter to permit specific objects to receive incoming connect requests sent to the cluster alias address. Section Section 3.9.1 describes how to identify DECnet–VAX objects.

3.3.3.1 Setting an Alias Node Identifier for the Executor

You establish an alias node identifier for the local node using the SET EXECUTOR command with the ALIAS NODE parameter. When the local node includes an alias node identifier in its database, it can be accessed by either the cluster alias or its individual node name or node address.

The alias node identifier can be either a node address or node name. Before you can establish a node name as a cluster alias, you must define the node name in the database, and associate it with a node address representing the whole VAXcluster, by means of the SET NODE or DEFINE NODE command. For example, the following command associates the node name CLUSTR with the address 2.13:

```
NCP>DEFINE NODE 2.13 NAME CLUSTR
```

You can then establish the name CLUSTR as the alias node identifier for the local node by using the following command:

```
NCP>DEFINE EXECUTOR ALIAS NODE CLUSTR
```

By entering these commands, you establish a node (CLUSTR) whose address is 2.13. This is the cluster alias node. Its address and name appear in the database like those of all other nodes. From the viewpoint of any node in the network outside the cluster, address 2.13, which is named CLUSTR, appears to be a real DECnet node that can participate in two-way communication. This cluster alias acts as a single node identifier that all participating nodes in the cluster can use to communicate with other nodes in the DECnet network.

3.3.3.2 Enabling Aliases for Nodes in a Cluster

When you manage the cluster alias node, you must decide whether participating nodes will accept incoming connect requests directed toward the cluster alias node identifier. You use the executor parameter ALIAS INCOMING to specify how incoming connect requests are to be handled. This parameter must be either enabled or disabled. To permit the node to accept incoming connect requests directed to the cluster alias node identifier, specify the ENABLED option. Otherwise, specify the DISABLED option to avoid receiving incoming connect requests directed to the cluster alias node identifier.

The following command prevents the local node from receiving incoming connect requests directed to the alias node identifier:

```
NCP>DEFINE EXECUTOR ALIAS INCOMING DISABLED
```

By default, the ALIAS INCOMING parameter is enabled for a node if an alias node identifier has been defined for the node.

3.3.4 Ethernet Addresses of Nodes

Nodes on Ethernet lines are identified by unique Ethernet addresses. A message can be sent to one, several, or all nodes on an Ethernet line simultaneously, depending on the Ethernet address used. You do not normally have to specify the Ethernet address of an individual node in order to configure your network; the software at the node sets its own Ethernet address. You need to know the Ethernet address of a node for service functions (such as downline load, circuit loopback test, and configurator operations) but not for normal network operations.

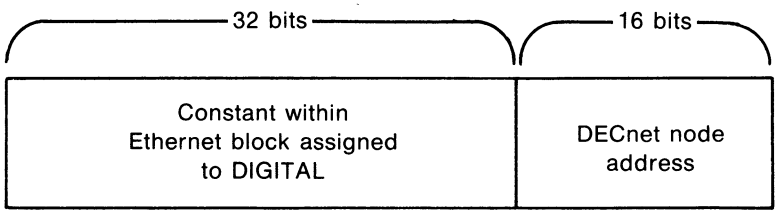
3.3.4.1 Format of Ethernet Addresses

An Ethernet address is 48 bits in length. Ethernet addresses are represented by six pairs of hexadecimal digits (6 bytes), separated by hyphens (for example, AA-01-23-45-67-FF). The bytes are displayed from left to right in the order in which they are transmitted; bits within each byte are transmitted from right to left. In the example, byte AA is transmitted first; byte FF is transmitted last.

Xerox Corporation assigns a block of addresses to a producer of Ethernet interfaces upon application. Thus, every manufacturer has a unique set of addresses to use. Normally, one address out of the assigned block of physical addresses is permanently associated with each interface (usually in a read-only memory). This address is known as the Ethernet **hardware address** of the interface.

DIGITAL's interface to Ethernet (for example, the DEUNA, DELUA, or DEQNA controller) can set a different address to be used by the interface; this address is known as the Ethernet **physical address**.

On powerup of the node, the Ethernet physical address is set to the hardware address. When DECnet starts an Ethernet line (for example, UNA-0), it causes the DEUNA connected to that line to set its physical address to be in the range of DIGITAL Ethernet addresses. As shown in the following figure, the DEUNA constructs the Ethernet physical address by appending the 16-bit executor node address to a constant 32-bit number (AA-00-04-00) within the block of Ethernet addresses assigned to DIGITAL.



ZK-1215-82

An example is a Phase IV routing node with DECnet address 1.182 (decimal), which would be set to an Ethernet physical address of AA-00-04-00-B6-04. Because the DEUNA at the node constructs its own physical address, users normally do not need to manipulate Ethernet addresses directly.

After the Ethernet physical address is set to its new value, it is reset to its original hardware address value only under the following circumstances:

- When a reset is issued to the DEUNA (for example, when the machine power is shut off)
- When the state of the Ethernet line is set to OFF

The Ethernet physical address of a node includes the number of the area in which the node resides. The area number is represented by the most significant 6 bits of the 16-bit DECnet node address, while the number of the node within the area is indicated by the least significant 10 bits of the node address (see Section 2.4.2). Therefore, changing the number of an area involves changing the Ethernet physical address of each node in that area.

If an existing network is not divided into areas, the default area number 1 is stored in the DECnet node address of each node. Conversion of an existing network to a multiple-area network may involve modification of the area number in the executor node address. During the conversion process (described in Section A.4), the network is shut down, the executor node address in the configuration database is modified to include the new area number, and the network is turned back on. When DECnet is restarted, it causes the DEUNA at the executor node to reset its Ethernet physical address.

However, if a non-DECnet communications application (such as a LAT terminal server) is connected to the same Ethernet as the executor node whose area number is being modified, you cannot change the Ethernet address of the executor used by the application while the interface to the application remains allocated. Thus, you should stop the application when DECnet is shut down and restart it after you turn DECnet back on.

3.3.4.2 Determining the Ethernet Physical Address of a Node

You can determine the Ethernet physical address of a node, as follows:

1. Convert the Phase IV node address (in the format *area-number.node-number*, as described in Section 3.7.2) to its decimal equivalent, using the following conversion algorithm:

   (area-number * 1024) + node-number

2. Convert the decimal node address to its hexadecimal equivalent, reversing the order of the bytes to form the hexadecimal node address.
3. Incorporate the hexadecimal node address in the following format:

   AA–00–04–00–hexnodeaddress

For example, to determine the Ethernet physical address of a node whose node address is 63.171, calculate the following:

(63 * 1024) + 171 = 64683 decimal = FCAB hexadecimal

This calculation causes the Ethernet physical address of the node to be the following:

```
AA-00-04-00-AB-FC
```

You can display the Ethernet physical address of a node by entering the following command:

```
NCP>SHOW EXECUTOR STATUS
```

The resulting display contains the Ethernet physical address of the executor, as follows:

```
Physical address          = AA-00-04-00-AB-FC
```

3.3.4.3 Ethernet Physical and Multicast Addresses

An Ethernet address can be a physical address of a single node or a multicast address, depending on the value of the low-order bit of the first byte of the address (this bit is transmitted first). The two types of node address are physical and multicast addresses.

The Ethernet **physical address** is the unique address of a single node on any Ethernet (as described previously). The least significant bit of the first byte of an Ethernet physical address is 0. (For example, in physical address AA–00–04–00–FC–00, byte AA in binary is 1010 1010 and the value of the low-order bit is 0.)

The Ethernet **multicast address** is a multidestination address of one or more nodes on a given Ethernet. The least significant bit of the first byte of a multicast address is 1. (For example, in the multicast address AB–22–22–22–22–22, byte AB in binary is 1010 1011 and the value of the low-order bit is 1.) A multicast address can be either of the following:

- **Multicast group address**. An address assigned to any number of nodes; you can use this address to send a message to all nodes in the group in a single transmission. The number of different groups that you can form equals the maximum number of multicast group addresses that you can assign.
- **Broadcast address**. A single multicast address (specifically, FF–FF–FF–FF–FF–FF) that you can use to transmit a message to all nodes on a given Ethernet. (Note that you should use the broadcast address only for messages to be acted upon by all nodes on the Ethernet, because all nodes must process them.)

3.3.4.4 Values of DIGITAL Ethernet Physical and Multicast Addresses

DIGITAL physical addresses are in the range AA–00–00–00–00–00 through AA–00–04–FF–FF–FF. Multicast addresses assigned for use in cross-company communications are as follows:

Value	Meaning
FF–FF–FF–FF–FF–FF	Broadcast
CF–00–00–00–00–00	Loopback assistance

DIGITAL multicast addresses assigned to be received by other DIGITAL nodes on the same Ethernet are as follows:

Value	Meaning
AB–00–00–01–00–00	Dump/load assistance
AB–00–00–02–00–00	Remote console
AB–00–00–03–00–00	All Phase IV routers
AB–00–00–04–00–00	All Phase IV end nodes
AB–00–00–05–00–00 through AB–00–03–FF–FF–FF	Reserved for future use
AB–00–04–00–00–00 through AB–00–04–FF–FF–FF	For use by DIGITAL customers for their own applications

DECnet always sets up the DEUNA at each node to receive messages sent to any address in the preceding list of DIGITAL multicast addresses. For information about how to send messages to Ethernet multicast addresses, refer to the *VMS I/O User's Reference Manual: Part II.*

3.3.5 Node Parameters

To establish information used to control various aspects of the local node's operation within the network, you specify the SET EXECUTOR command. You can set several parameters with the SET EXECUTOR command. You must specify the parameter ADDRESS. You should also specify MAXIMUM ADDRESS, BUFFER SIZE, and TYPE. (If the node is an end node, then you can use the default node type.)

In addition, you may want to specify names, access control information, and node counter event logging information for any or all of the remote nodes in your network. If a remote node can be loaded downline, you can specify a number of default parameters to be used locally to perform a downline load or upline dump operation. Table 3–1 lists all node parameters by function and indicates whether they apply to local or remote nodes or to both. Note that this table refers to the local node's definition of its executor parameters and the local node's definition of its remote nodes.

Table 3–1 Node Parameters and Their Functions

Parameters According to Function	Executor Node	Remote Node
Node identification		
ADDRESS node-address	X	X
ALIAS NODE	X	
IDENTIFICATION id-string	X	
NAME node-name	X	X
Loop node identification		
CIRCUIT circuit-id		X

Table 3–1 (Cont.) Node Parameters and Their Functions

Parameters According to Function	Executor Node	Remote Node
Counter timing for node counter logging events		
COUNTER TIMER seconds	X	X
Local node state		
STATE	X	
ON		
OFF		
RESTRICTED		
SHUT		
Access control		
ACCESS		X
INCOMING		
OUTGOING		
BOTH		
NONE		
DEFAULT ACCESS	X	
INCOMING		
OUTGOING		
BOTH		
NONE		
INCOMING PROXY	X	
ENABLED		
DISABLED		
NONPRIVILEGED	X	X
ACCOUNT account		
PASSWORD password		
USER user-id		
OUTGOING PROXY	X	
ENABLED		
DISABLED		
PRIVILEGED	X	X
ACCOUNT account		
PASSWORD password		
USER user-id		

Table 3–1 (Cont.) Node Parameters and Their Functions

Parameters According to Function	Executor Node	Remote Node
Downline loading		
CPU cpu-type		X
DIAGNOSTIC FILE file-spec		X
HARDWARE ADDRESS E-address		X
HOST node-id		X
LOAD ASSIST AGENT file-spec		X
LOAD ASSIST PARAMETER item		X
LOAD FILE file-spec		X
MANAGEMENT FILE file-spec		X
SECONDARY LOADER file-spec		X
SERVICE CIRCUIT circuit-id		X
SERVICE DEVICE device-type		X
SERVICE PASSWORD hex-password		X
SOFTWARE IDENTIFICATION software-id		X
SOFTWARE TYPE type		X
TERTIARY LOADER file-spec		X
Upline dumping		
DUMP ADDRESS number		X
DUMP COUNT number		X
DUMP FILE number		X
Routing initialization passwords		
RECEIVE PASSWORD password		X
TRANSMIT PASSWORD		X
DDCMP circuit connection control		
INBOUND node-type		X
Data link control		
BUFFER SIZE number	X	
MAXIMUM ADDRESS number	X	
MAXIMUM BUFFERS number	X	
MAXIMUM CIRCUITS number	X	
Logical link control		
ALIAS MAXIMUM LINKS number	X	
DELAY FACTOR number	X	
DELAY WEIGHT number	X	
INACTIVITY TIMER seconds	X	
INCOMING TIMER number	X	
MAXIMUM LINKS number	X	
OUTGOING TIMER seconds	X	
PIPELINE QUOTA quota	X	
RETRANSMIT FACTOR number	X	
SEGMENT BUFFER SIZE	X	

Table 3–1 (Cont.) Node Parameters and Their Functions

Parameters According to Function	Executor Node	Remote Node
Routing control		
ALIAS INCOMING option	X	
AREA MAXIMUM COST number	X	
AREA MAXIMUM HOPS number	X	
BROADCAST ROUTING TIMER seconds	X	
MAXIMUM AREA number	X	
MAXIMUM BROADCAST NONROUTERS number	X	
MAXIMUM BROADCAST ROUTERS number	X	
MAXIMUM COST number	X	
MAXIMUM HOPS number	X	
MAXIMUM PATH SPLITS number	X	
MAXIMUM VISITS number	X	
PATH SPLIT POLICY policy	X	
ROUTING TIMER seconds	X	
TYPE node-type	X	
	X	
Incoming X.25 call control		
SUBADDRESSES range	X	

You can specify parameters common to both the local node and remote nodes in one of two ways: use either the SET EXECUTOR or SET NODE command with the local node name or address to establish or modify parameters for the local node, or use the SET NODE command to establish or modify parameters for a remote node.

When using either the SET EXECUTOR or SET NODE command to establish or modify parameters for the local node, be sure to avoid using, in the same command, parameters listed only under the Local Node column and those listed under both the Local Node and Remote Node columns in Table 3–1. If you mix these parameters in a single command, you receive an "invalid parameter grouping" message. For example, the following commands are valid:

```
NCP>SET EXECUTOR ADDRESS 11
NCP>SET NODE 11 NAME BOSTON
NCP>SET NODE 15 SERVICE CIRCUIT DMC-1 -
_ RECEIVE PASSWORD RSXNODE
```

The first command specifies a node address for the local node. The second command specifies a node name for the node whose address is 1.11, which in this case is the local node. The third command specifies parameters for node 1.15. Note that this command contains parameters for both remote and adjacent nodes. The following command, however, is invalid because it mixes parameters:

```
NCP>SET EXECUTOR ADDRESS 11 NAME BOSTON
%NCP-W-INVPGP, Invalid parameter grouping, Address
```

The CLEAR NODE command clears parameters for either the local or remote nodes. The CLEAR EXECUTOR command clears local node parameters common to both local and remote nodes. You cannot clear the local node parameters BUFFER SIZE and STATE from the volatile database. You can, however, purge them from the permanent database with the PURGE

EXECUTOR command. You can use this same command to remove local node parameters common to both local and remote nodes from the permanent database.

After the local node's state is set to ON, you cannot change the ADDRESS or BUFFER SIZE parameters for the local node.

3.3.5.1 Data Link Control

Several local node parameters regulate various aspects of physical line operation. Specify the size of NSP receive buffers and transmit buffers (segment buffers), the number of buffers used to transmit on all circuits, and the number of circuits that the local node can use as DECnet–VAX communication lines. NCP provides four parameters for this purpose: BUFFER SIZE, SEGMENT BUFFER SIZE, MAXIMUM BUFFERS, and MAXIMUM CIRCUITS. You should be careful to set the values for these parameters to a reasonable level, or system performance may suffer. The parameters all have reasonable default values.

For X.25 data links, the node parameters do not directly affect physical line operation.

Setting Buffer Sizes

To specify the maximum size (in bytes) of NSP receive buffers, use the BUFFER SIZE parameter. For example, the following command sets the size of all receive buffers for the executor node to 576 bytes:

```
NCP>SET EXECUTOR BUFFER SIZE 576
```

This parameter also controls the maximum segment size of all NSP messages that the local node can receive and forward. The buffer size value that you select is used for all lines. You cannot use the BUFFER SIZE parameter to select individual values for individual lines. You can, however, use the BUFFER SIZE parameter in the SET LINE command to override the BUFFER SIZE value set in the executor for specific logical links on a line, such as a particular Ethernet line (see Section 3.6.2.2).

The default BUFFER SIZE value is equal to the SEGMENT BUFFER SIZE if specified; otherwise, the default size is 576 bytes. At a minimum, the buffer size must be 192 bytes. For more information in this area, refer to Chapter 5 and to the *Network Services Protocol Functional Specification.*

You should consider a number of things when selecting the buffer size value.

- These buffers require nonpaged memory pool.
- The SYSGEN parameter LRPSIZE should be set to the executor buffer size. In addition, the parameter LRPCOUNT should be at least as large as the total number of all receive buffers on all lines, plus twice the number of circuits.
- Faster lines perform better with large buffers and large user messages that reduce the processor load. (The smaller the segments, the more messages are processed.)
- Lines that are error prone (for example, telephone lines) should use small buffers (256 bytes) to reduce both the probability and the cost of retransmissions.

The procedure for changing buffer sizes is described next.

Note: Using the same buffer size for all nodes in your network is strongly recommended. Otherwise, nodes with smaller buffer sizes drop packets when you attempt to route through them.

Changing Buffer Sizes

You can change the size of the buffers on all nodes without bringing down the entire network by performing a two-pass conversion process involving a second parameter, the SEGMENT BUFFER SIZE, as well as the BUFFER SIZE parameter. The conversion process requires only that you set the local node to the OFF state while changing the buffer size.

The maximum size of the transmit buffer is specified in the SEGMENT BUFFER SIZE parameter. For example, the following command sets the maximum size of a transmit buffer to 576 bytes:

```
NCP>SET EXECUTOR SEGMENT BUFFER SIZE 576
```

The maximum size of the receive buffer is specified in the BUFFER SIZE parameter. The following command sets the maximum size of the receive buffer to 576 bytes:

```
NCP>SET EXECUTOR BUFFER SIZE 576
```

The SEGMENT BUFFER SIZE normally has the same value as the BUFFER SIZE, but may be set to less in order to perform the buffer size conversion process. The default value of the SEGMENT BUFFER SIZE is equal to the BUFFER SIZE if specified; otherwise, the default size is 576 bytes.

The steps in the conversion depend on whether you are increasing or decreasing the size of the buffers. To increase the size of the buffers, perform the following conversion:

1. Reset the value of the BUFFER SIZE parameter at each node to the larger size, permitting each node to receive a larger message.
2. Reset the value of the SEGMENT BUFFER SIZE parameter at each node to the larger size, permitting each node to transmit a larger message.

This two-step process ensures that all nodes are able to receive and forward larger messages before any node is able to transmit a larger message.

To decrease the size of the buffers, perform the following conversion:

1. Reset the value of the SEGMENT BUFFER SIZE parameter at each node to the smaller size, decreasing the size message that each node can transmit.
2. Reset the value of the BUFFER SIZE parameter at each node to the smaller size, decreasing the size message all nodes can receive.

This process ensures that the size of messages that can be transmitted across the network is decreased before the size of the buffers that receive and forward messages is decreased.

An example of the conversion process involves increasing the size of messages that can be transmitted and received over the network from 576 bytes to 1000 bytes. First, enter the following command at each node in the network:

```
NCP>SET EXECUTOR BUFFER SIZE 1000
```

Then, enter the following command at each node in the network:

```
NCP>SET EXECUTOR SEGMENT BUFFER SIZE 1000
```

Each node will first be able to receive and forward 1000-byte messages, and then will be able to transmit them.

Maximum Number of Buffers

To specify the maximum number of buffers for the transmit buffer pool, use the MAXIMUM BUFFERS parameter. The value you assign determines the size of internal data structures for DECnet–VAX software. For example, the following command sets the maximum number of buffers to 20:

```
NCP>SET EXECUTOR MAXIMUM BUFFERS 20
```

If you do not specify a value for this parameter, DECnet–VAX provides a default value of 100. Thus, you do not have to specify a value unless you want to limit the amount of nonpaged pool used by DECnet–VAX. For most operations, DECnet–VAX allocates only as many buffers as it needs (even if you specify a greater number than the amount needed), and does not allocate more than the number of buffers you specify.

Maximum Number of Circuits

To specify the maximum number of circuits that the user can identify in the volatile database, use the MAXIMUM CIRCUITS parameter. This value determines the size of internal data structures for DECnet–VAX software. For example, the following command establishes an upper limit of 3 circuits that the local node can use:

```
NCP>SET EXECUTOR MAXIMUM CIRCUITS 3
```

The default value for this parameter is 16.

If the local node is connected to an Ethernet circuit, you cannot change the value of the MAXIMUM CIRCUITS parameter while the executor and the Ethernet circuit are in the ON state.

3.3.5.2 Operational State of the Local Node

Use the STATE parameter in conjunction with the SET EXECUTOR command to exercise control of the operational state of the local node. There are four states associated with this parameter:

OFF	Allows no new logical links to be created, terminates existing links, and stops route-through traffic. The NETACP process exits.
ON	Allows new logical links to be created. The ON state is the normal operational state allowing route-through traffic.
RESTRICTED	Allows no new logical links from other nodes, yet does not inhibit route-through.
SHUT	Similar to RESTRICTED. However, after all logical links are terminated, the local node goes to the OFF state.

Any network user with the OPER privilege can initiate or confirm a logical link connection even though the local node is in the RESTRICTED or SHUT state. Thus, a system manager can use NCP and NML when the network is in either of these states.

Chapter 6 discusses local node states in terms of controlling the operation of the local node, and thus the network as a whole.

Note: You should never use the DEFINE EXECUTOR STATE OFF command for the permanent database because this command would cause the network to shut down immediately after being started.

3.3.6 Copying Node Databases

You can update your node database by copying current information about remote nodes from the configuration database of any node to which you have access. Use the NCP command COPY KNOWN NODES to copy the volatile or permanent node database entries from a remote node to either or both the volatile and permanent node databases on your node. If you specify the WITH CLEAR or WITH PURGE qualifier on the COPY command, the local node database to which the information is to be copied is cleared or purged before the information is copied. Only the executor node characteristics and the name and address of the remote node are retained when the database is cleared or purged before a copy operation.

The COPY KNOWN NODES command permits you to update your existing node database to reflect current data on remote nodes without having to shut down your node. If your network is large, the COPY command provides you with a means of keeping up with frequent changes in the composition of the network. For example, one node on an Ethernet may serve as the master, keeping in its node database current information on all remote nodes that can be accessed over the network. Then, as other nodes come up on the same Ethernet, they can obtain the latest version of the node database by copying it from the master node.

If you did not specify any remote node entries when you configured your node, you can use the COPY command at any time to obtain the remote node entries that indicate the nodes to which you have access. If you want to copy the node database from a remote node that is not defined in your volatile database, you can specify the node's address in the COPY command; execution of the copy operation causes the name and address of the node to be defined in your database.

You cannot use COPY KNOWN NODES to copy a subset of a node database.

An alternative method for copying node database information is to use the DCL COPY command to copy the existing permanent database file located in SYS$SYSTEM:NETNODE_REMOTE.DAT, as described in Section 3.3.6.5.

3.3.6.1 COPY Command Parameters and Qualifiers

The COPY KNOWN NODES command causes the names and addresses of remote nodes to be copied from a remote database to the local node database or databases you specify. The FROM *node-id* parameter in the COPY command specifies the remote node from which the node database information is to be copied.

You can include explicit access control information in the node-id field, or default to a proxy or DECnet account on the remote node, as appropriate. The following command copies remote node data from node BOSTON on which user BROWN has an account with the password PASS123:

```
NCP>COPY KNOWN NODES FROM BOSTON"BROWN PASS123"
```

To indicate which node database at the remote node is to be copied, specify the USING VOLATILE or USING PERMANENT qualifier. If you do not specify the USING qualifier, the default value is USING VOLATILE.

You can indicate the local node database to which the information is to be copied by specifying the TO qualifier. The local node database can be the volatile or permanent database, or both. In the following command, the volatile database at node BANGOR is to be copied to both the volatile and permanent databases at the local node:

```
NCP>COPY KNOWN NODES FROM BANGOR USING VOLATILE TO BOTH
```

If you do not specify the TO qualifier, the local database defaults to VOLATILE.

To clear or purge the local database before the copy operation is performed, specify the WITH CLEAR or WITH PURGE qualifier in the COPY command. The WITH CLEAR qualifier is appropriate if the local database is volatile; the WITH PURGE qualifier is appropriate if it is permanent. Specify either WITH CLEAR or WITH PURGE if both the volatile and permanent databases are to be cleared. (In practice, you can specify either value of the WITH qualifier to clear or purge either or both of the local databases.) The following command indicates that the permanent node database at the local node is to be purged before the remote node data from node BANGOR is copied to the local database:

```
NCP>COPY KNOWN NODES FROM BANGOR USING PERMANENT -
_ TO PERMANENT WITH PURGE
```

3.3.6.2 Clearing and Purging the Local Node Database

During a copy operation, if the volatile database is to be cleared, the entries for the executor node and any loop nodes are not cleared. If the permanent database is to be purged, however, the entry for the executor node is purged. Therefore, before the purge occurs, the copy operation causes the executor node characteristics to be saved: a LIST EXECUTOR CHARACTERISTICS command is executed and the executor characteristics are stored. After the purge is executed, the executor characteristics are reinserted in the local node database.

In addition, the local node must retain the name and address of the remote node from which it is to copy the data. Before the clear or purge operation is performed, the name and address of the remote node are saved. This information is reinserted in the local node database after the clear or purge operation is completed. Thus, to purge your database without purging your executor database, use the COPY KNOWN NODES command with the WITH PURGE qualifier.

Note that if the executor or remote node is not defined in the local node database, an error results.

If an error occurs during execution of the LIST EXECUTOR CHARACTERISTICS command, the purge is aborted. After the LIST operation is performed, purging continues even if errors are encountered.

Clearing or purging the local database as part of a copy operation is recommended. If a clear or purge operation is not performed before the remote node data is copied, conflicts can occur between original node entries in the local database and node entries being copied from the remote node database. A node must be identified uniquely, by one name and one address. A conflict exists when the entries being copied on an existing database would cause one node address to be associated with two node names or two node

addresses with one node name. When such a conflict occurs during the copy operation, the original node entry is not updated and an informational message is displayed. For example, if the local node database identified node A with address 3.1 and node B with address 3.3, an attempt to copy an entry that defines node A with address 3.3 would fail, and an informational message would be issued.

3.3.6.3 Copying the Node Database from a Remote Node

Entering a COPY KNOWN NODES command accomplishes the following tasks:

- If you indicate that the volatile node database at the remote node is to be copied (by specifying the USING VOLATILE qualifier in the COPY command), a SHOW KNOWN NODES command is executed at the remote node. If you indicate that the permanent node database is to be copied (by specifying the USING PERMANENT qualifier), a LIST KNOWN NODES command is executed at the remote node.
- The COPY operation extracts remote node names and addresses from data returned by the SHOW or LIST command.
- For each node name and address extracted, a SET or DEFINE NODE command is executed on the appropriate local node database. If you indicate in the COPY command that the information is to be copied to the volatile database, the following command is executed for each entry:

  ```
  SET NODE ADDRESS address NAME name
  ```

  If you indicate that the information goes to both local node databases, both SET NODE and DEFINE NODE commands are executed for each remote node entry. When the COPY operation receives the name and address of the local node, no SET or DEFINE command is performed.

When the name and address of the remote node from which the data is being copied is returned, the entry indicates that it is the executor node. When the remote node is defined in the local database, however, it is not listed as an executor node. Loop node names listed in the node database at the remote node are not defined in the local database.

After the COPY operation begins defining the remote nodes, it continues trying to define the nodes despite any errors it may encounter. It displays informational messages for errors in individual node entries.

3.3.6.4 Example of Copying Remote Node Data

The following examples illustrate how to use the COPY command to copy remote node entries from the permanent node database at node ROBIN to the permanent node database at node LARK without purging the local node database. In this example, node LARK has not defined the executor node or remote node ROBIN in its database; therefore, error messages are displayed. Note that the copy operation is not performed for nodes A and C, because of a conflict between existing and updated addresses for these nodes. Informational messages display for the entries for nodes A and C. (In the first example, only the node entries resulting from the LIST commands are displayed.)

To determine the node entries on the permanent node database at the local node, enter the following command, which causes the node entries to be displayed. Note that, in this example, the executor is not defined.

```
NCP>LIST KNOWN NODES

    Remote node = 2.1 (C)
    Remote node = 2.3 (A)
```

You can determine the node entries in the permanent node database at remote node ROBIN (whose address is 2.20) by entering the following command, which causes the node entries to be displayed. You can reach node ROBIN by specifying its address in the NCP command, even though the node is not listed by name in the local node database.

```
NCP>TELL 2.20 LIST KNOWN NODES

    Executor node = 2.20 (ROBIN)
    Remote node = 2.1 (A)
    Remote node = 2.2 (B)
    Remote node = 2.3 (C)
```

To perform the copy operation, enter the following COPY command:

```
NCP>COPY KNOWN NODES FROM 2.20 USING PERMANENT

%NCP-W-UNRCMP, Unrecognized component, Node
%NCP-W-EXEABO, Executor characteristics not defined.
%NCP-W-UNRCMP, Unrecognized component, Node
%NCP-W-REMABO, Remote node not defined.
%NCP-W-INVPVA, Invalid parameter value, Address
Remote node = 2.1 (C)

NODE  2.1    NAME A

%NCP-W-INVPVA, Invalid parameter value, Address
Remote node = 2.3 (A)

NODE  2.3    NAME C
```

The error messages generated during the copy operation are displayed on your screen directly under the COPY command. Note that remote node entries successfully copied to the local node database (such as node B) are not displayed under the COPY command. The COPY command ignores these error messages because they do not affect the copy operation.

To determine the final results of the copy operation, enter the LIST KNOWN NODES command at your node to obtain the following display of node entries:

```
NCP>LIST KNOWN NODES

    Remote node = 2.1 (C)
    Remote node = 2.2 (B)
    Remote node = 2.3 (A)
    Remote node = 2.20 (ROBIN)
```

3.3.6.5 Copying the Permanent Node Database Using DCL COPY

Rather than using the NCP command COPY KNOWN NODES to copy node database information, you can use the DCL COPY command to obtain the contents of an existing permanent node database residing on a remote node in the file SYS$SYSTEM:NETNODE_REMOTE.DAT. You must have the required privileges on the remote node to access this file.

When you enter the DCL COPY command to copy from the remote node, you must include the remote node-id in the file specification. You can include explicit access control information in the node-id field, or default to a proxy or DECnet account on the remote node, as appropriate. The following command copies remote node data from node BOSTON on which the user SYSTEM has a proxy account.

```
$ COPY BOSTON::SYS$SYSTEM:NETNODE_REMOTE.DAT *
```

This COPY command copies the permanent node database from the remote node, replacing your local database file. After you copy this permanent node database file to your local node, you can enter the node database information into the local NCP volatile database by entering the following SET NODE command:

```
NCP>SET KNOWN NODES ALL
```

3.3.7 Node Counters

DECnet software automatically collects certain statistics for nodes in the network. These statistics are known as node counters. Such information may include the number of connects sent and received, the number of messages sent and received, and the number of packets lost. This information may be useful either alone or in conjunction with logging information to evaluate the performance of a given node. The network counter summary in the *VMS Network Control Program Manual* describes the complete list of node counters. Refer to Section 2.9 for a discussion of logging.

You can use NCP to regulate how often counters are logged and when they are zeroed. To do so, you can use the SET EXECUTOR or the SET NODE command with the COUNTER TIMER parameter. For example, the following command causes a node counter logging event to take place every 600 seconds for the local node:

```
NCP>SET EXECUTOR COUNTER TIMER 600
```

The counters are then zeroed. Similarly, the following command specifies that counters for remote node TRNTO are to be logged at the local node every 600 seconds:

```
NCP>SET NODE TRNTO COUNTER TIMER 600
```

Note that these counters are maintained on the local node. To clear the COUNTER TIMER parameter, use the CLEAR EXECUTOR or CLEAR NODE command along with this parameter.

You can display node counter statistics at any time while the network is running by using the SHOW NODE COUNTERS command.

In addition, at any point when the network software is running, you can zero node counters for a given remote node, the local node, or for all known nodes. Use any of the following commands to zero node counters:

```
NCP>ZERO EXECUTOR COUNTERS
NCP>ZERO NODE BOSTON COUNTERS
NCP>ZERO KNOWN NODES COUNTERS
```

3.4 X.25 Protocol Module Commands

The X25–PROTOCOL module implements the X.25 level 3 protocol, which controls the transmission of data packets. This module structures control and user data into packets, sequences these packets for transmission, and establishes, maintains, and clears X.25 virtual circuits. This module also associates PSDNs, local DTE addresses, and optionally group names, with this controlling information.

Use separate SET MODULE X25–PROTOCOL commands to specify a network, a DTE, and a group.

3.4.1 Network Identification

Use the NETWORK qualifier to identify a network (that connects to a PSDN). The network name must be unique both on this database and any X25-ACCESS databases.

You must specify a *profile* as a parameter to the NETWORK qualifier. The profile is a file that gives maximum, minimum, and default values for PSDN parameters. The profile also gives the facilities of the PSDN, such as FAST SELECT.

The network profile is set up as part of the FCNS (Field Configurable Network Software) procedure, when VAX PSI is first matched with a particular PSDN. For further details, refer to the VAX PSI documentation set.

The following command associates with the X25-PROTOCOL module the network TELENET1, which has the profile TELENET:

```
NCP>SET MODULE X25-PROTOCOL NETWORK TELENET1 PROFILE TELENET
```

3.4.2 Local DTE Identification

Use the DTE qualifier followed by a NETWORK parameter to identify a local DTE and the network to which it is connected. For example:

```
NCP>SET MODULE X25-PROTOCOL DTE 123789456 NETWORK TELENET1 ...
```

This command specifies a DTE that has the address 123789456 in network TELENET1.

Refer to the *Public Network Information* manual for DTE address formats.

3.4.2.1 Operational State of DTE

The STATE parameter specifies one of three operational states for each DTE:

OFF	Prevents use of the DTE and clears all existing virtual circuits.
ON	Allows normal use of the DTE.
SHUT	Prevents use of the DTE for any new activity, but allows existing virtual circuits to complete their operation.

The following command allows normal use of the DTE:

```
NCP>SET MODULE X25-PROTOCOL DTE 123789456 NETWORK TELENET1 STATE ON ...
```

The ON state has substates. For a complete list of states, substates, and their transitions, refer to the *VMS Network Control Program Manual*. The STATE parameter is optional. If you do not specify it, the state is set to ON.

3.4.2.2 Line Identification

The LINE parameter identifies the line associated with each DTE. Each DTE must have a unique line. Line identification takes the following format:

dev-c[-u]

where:

dev	Is a device name.
c	Is a decimal number designating the device's hardware controller.
u	Is a decimal unit or line number included if the device is a multiple unit line controller.

Section 3.6.1 describes line identification.

Note that the unit number is optional. The following command identifies the line KMX-0-0 associated with DTE 123789456:

```
NCP>SET MODULE X25-PROTOCOL DTE 123789456 NET TELENET1 -
_ LINE KMX-0-0 ...
```

Use the SET LINE command to specify parameters for this line.

When you specify a DTE for the first time, the LINE parameter is mandatory.

3.4.2.3 Channel Identification

The CHANNELS parameter associates a set of **logical channels** to be used for outgoing calls with each DTE. Outgoing calls are all calls that originate from your DTE. Specify one or more **logical channel numbers (LCNs)** as a list. Separate multiple LCNs with hyphens to indicate ranges, and commas to indicate individual numbers. For example, the following command indicates that 20 is the first LCN, counting down from 20 to 10, then 3, and finally 9:

```
NCP>SET MODULE X25-PROTOCOL DTE 123789456 -
_ CHANNELS 20-10,3,9 ...
```

Specify a value in the range 1 to 4095 for each number in the range.

The values you specify are those supplied by the PSDN authorities at subscription time.

3.4.2.4 MAXIMUM CIRCUITS Parameter

The MAXIMUM CIRCUITS parameter specifies the maximum number of circuits that each DTE can handle. For example, the following command specifies that the DTE 123789456 can handle a maximum of 200 circuits:

```
NCP>SET MODULE X25-PROTOCOL DTE 123789456 -
_ MAXIMUM CIRCUITS 200...
```

The MAXIMUM CIRCUITS parameter is optional and, by default, the maximum is 512.

When you specify a DTE for the first time, this parameter indicates the size of the control area allocated from nonpaged pool. Thus, you cannot increase this value while the software is running. However, you can decrease the value and increase it again, provided that you do not specify a value larger than the original set. If this value is larger than required, nonpaged pool will be wasted.

3.4.2.5 INTERFACE Parameter

The INTERFACE parameter specifies whether VAX PSI operates as a DTE, DCE, or automatically selects the correct interface (DTE or DCE).

For example, the following command specifies that VAX PSI is to operate as a DCE:

```
NCP>SET MODULE X25-PROTOCOL DTE 123789456 INTERFACE DCE
```

This parameter is of use only with the ISO8208 network profile. See also Section 3.4.8.

3.4.3 Data Packet Control

The transmission of data packets over an X.25 virtual circuit is determined by the size of the packet and the window.

3.4.3.1 Packet Size

The MAXIMUM DATA parameter specifies the maximum size of packets for all X.25 virtual circuits. The following command sets the maximum packet size to 128 bytes:

```
NCP>SET MODULE X25-PROTOCOL DTE ... MAXIMUM DATA 128 ...
```

The packet size must always be at least 5 bytes smaller than the maximum size of the **frame** on a line (see Section 3.6.5). If the value in the PSI$_NCB_PKTSIZE field of the network connect block (NCB) is greater than the value you specified with the MAXIMUM DATA parameter, the MAXIMUM DATA value is used.

The DEFAULT DATA parameter specifies a default packet size for all X.25 virtual circuits. For example, the following command sets the default packet size to 64 bytes:

```
NCP>SET MODULE X25-PROTOCOL DTE ... DEFAULT DATA 64 ...
```

The default packet size must always be less than or equal to the maximum packet size. If you do not specify the PSI$_NCB_PKTSIZE field of the NCB, the default packet size is used. The value must be a power of 2 in the range 16 to 4096.

The MAXIMUM DATA and DEFAULT DATA parameters are optional and take the network value by default; otherwise, you must set them to the values specified in your PSDN subscription. See the *Public Network Information* manual for the network values of these parameters.

3.4.3.2 Window Size

The MAXIMUM WINDOW parameter specifies the maximum window size for all X.25 virtual circuits. The value for this parameter is the maximum number of packets for which outstanding acknowledgments are allowed. The following command sets the maximum window size to 3:

```
NCP>SET MODULE X25-PROTOCOL DTE ... MAXIMUM WINDOW 3 ...
```

If the value in the PSI$_NCB_WINSIZE field of the NCB is greater than the value you specified using the MAXIMUM WINDOW parameter, the MAXIMUM WINDOW value is used.

The DEFAULT WINDOW parameter specifies a default window size for all X.25 virtual circuits. For example, the following command sets the default window size to 2:

```
NCP>SET MODULE X25-PROTOCOL DTE ... DEFAULT WINDOW 2 ...
```

The default window size must always be less than or equal to the maximum window size. If you do not specify the PSI$_NCB_WINSIZE field of the NCB, the default window size is used.

Specify values in the range 1 to 7 (unless extended sequence numbering is used on the PSDN, in which case the range is 1 to 127). The values must agree with your PSDN subscription.

The MAXIMUM WINDOW and DEFAULT WINDOW parameters are optional and, by default, take the network value. See the *Public Network Information* manual for the network values of these parameters.

Note that these parameters affect only the X.25 level 3 window size. They are independent of the X.25 level 2 window size, which is controlled by the line parameter MAXIMUM WINDOW (see Section 3.6.5).

3.4.4 Call Request Packet Control

Use the CALL TIMER parameter to control call setup. This timer starts to run when a request to set up a virtual circuit is transmitted; if it expires before a response has been received, the request is cleared. In the following command, the request to set up a virtual circuit is cleared if no response has been received within 10 seconds:

```
NCP>SET MODULE X25-PROTOCOL DTE ... CALL TIMER 10 ...
```

Specify a value in the range 1 to 255.

The CALL TIMER parameter is optional and, by default, takes the network value. See the *Public Network Information* manual for the network value of this parameter.

3.4.5 Clear Request Packet Control

Two parameters control transmission of clear request packets over SVCs: CLEAR TIMER and MAXIMUM CLEARS.

Use the CLEAR TIMER parameter to control how often clear request packets are retransmitted if not acknowledged. The following command sets the retransmission frequency to 20 seconds:

```
NCP>SET MODULE X25-PROTOCOL DTE ... CLEAR TIMER 20 ...
```

Specify a value in the range 1 to 255.

Use the MAXIMUM CLEARS parameter to specify the maximum number of times a clear request packet is retransmitted over the circuit. For example, the following command indicates that if a clear request is not acknowledged within 20 seconds, the request is retransmitted, and that this operation is to be performed a maximum of 8 times:

```
NCP>SET MODULE X25-PROTOCOL DTE ... CLEAR TIMER 20 -
_ MAXIMUM CLEARS 8 ...
```

The circuit is assumed to be cleared if the clear request is still not acknowledged by this time.

Specify a value in the range 1 to 255.

The CLEAR TIMER and MAXIMUM CLEARS parameters are optional and take the network value by default. See the *Public Network Information* manual for the network values of these parameters.

3.4.6 Reset Control

Two parameters control transmission of reset packets over SVCs and PVCs: RESET TIMER and MAXIMUM RESETS.

Use the RESET TIMER parameter to control how often reset packets are retransmitted if not acknowledged. For example, the following command sets the retransmission frequency to 10 seconds:

```
NCP>SET MODULE X25-PROTOCOL DTE ... RESET TIMER 10 ...
```

Specify a value in the range 1 to 255.

Use the MAXIMUM RESETS parameter to specify the maximum number of times a reset is retransmitted to the DCE. The following command indicates that if a reset is not acknowledged within 10 seconds, the reset is retransmitted, and that this operation is to be performed a maximum of 8 times:

```
NCP>SET MODULE X25-PROTOCOL DTE ... RESET TIMER 10 -
_ MAXIMUM RESETS 8 ...
```

The circuit is cleared if the reset is still not acknowledged by this time. Specify a value in the range 1 to 255.

The RESET TIMER and MAXIMUM RESETS parameters are optional and take the network value by default. See the *Public Network Information* manual for the network values of these parameters.

3.4.7 Restart Control

Two parameters control transmission of restart packets over the data link to the DCE: RESTART TIMER and MAXIMUM RESTARTS. Use the RESTART TIMER to control how often restart packets are retransmitted. For example, the following command sets the retransmission frequency to 20 seconds:

```
NCP>SET MODULE X25-PROTOCOL DTE ... RESTART TIMER 20 ...
```

Specify a value in the range 1 to 255.

Use the MAXIMUM RESTARTS parameter to specify the maximum number of times a restart is retransmitted to the DCE. The following command specifies that a restart is to be retransmitted every 20 seconds, and that this operation is to be performed a maximum number of 10 times:

```
NCP>SET MODULE X25-PROTOCOL DTE ... RESTART TIMER 20 -
_ MAXIMUM RESTARTS 10 ...
```

Specify a value in the range 1 to 255.

The RESTART TIMER and MAXIMUM RESTARTS parameters are optional and take the network value by default. See the *Public Network Information* manual for the network values of these parameters.

3.4.8 ISO Networks

There are two parameters provided for use with an International Standard 8208 packet switching network. (8208 is the International Standards Organization's definition of the CCITT X.25 recommendations.)

The two parameters are INTERFACE and INTERRUPT TIMER. INTERFACE allows you to specify that your VAX PSI system acts in one of three ways: as a DTE, as a DCE, or automatically as either a DTE or DCE (parameter value NEGOTIATED). Refer to Section 3.4.2.5 for an example of specifying the INTERFACE parameter.

INTERRUPT TIMER controls how long an interrupt may remain unconfirmed. If the interrupt is not confirmed within this time, the circuit is reset. The following command sets the interrupt timer to 150 seconds:

```
NCP>SET MODULE X25-PROTOCOL DTE ... INTERRUPT TIMER 150
```

3.4.9 Group Identification

If you are a member of either a closed user group (CUG) or bilateral closed user group (BCUG), always identify the group with the GROUP qualifier. Each group should have a unique name, which is a string 2 to 16 characters long. The following commands show a series of group specifications:

```
NCP> SET MODULE X25-PROTOCOL GROUP ESECUG ...
NCP> SET MODULE X25-PROTOCOL GROUP DECCUG ...
NCP> SET MODULE X25-PROTOCOL GROUP BASINGSTOKE ...
       .          .          .          .
       .          .          .          .
       .          .          .          .
```

The local DTE, the group type, and the group number should be associated with each group.

3.4.9.1 Local DTE Identification

Use the DTE parameter to specify the address of the DTE associated with the group name, and the NETWORK parameter to identify the network to which the DTE belongs. For example:

```
NCP>SET MODULE X25-PROTOCOL GROUP ESECUG -
_ DTE 123789456 NETWORK PSS1 ...
```

When you specify a group for the first time, the DTE parameter is mandatory. Note that after you set parameters for a group, you cannot change them except by clearing the group and setting it up again.

3.4.9.2 Group Number

Use the NUMBER parameter to specify the number that identifies your group. These numbers are allocated by the PSDN at subscription time. For example:

```
NCP>SET MODULE X25-PROTOCOL GROUP ESECUG NUMBER 12 ...
```

When you specify a group for the first time, the NUMBER parameter is mandatory.

3.4.9.3 Group Type

If you are a member of a BCUG, use the TYPE BILATERAL parameter to specify this group type. For example:

```
NCP>SET MODULE X25-PROTOCOL GROUP DECCUG TYPE BILATERAL ...
```

If the group is a CUG, omit this parameter.

3.4.10 X.25 Protocol Module Counters

VAX PSI automatically maintains certain statistics for the X25-PROTOCOL module in the network. These statistics are known as protocol module counters. They may include the number of bytes, data blocks, calls, and fast selects sent and received; the number of active channels; the number of resets sent, received, or initiated by the network; and the number of restarts. These statistics are useful in monitoring the activity of the component. A complete list of protocol module counters is provided in the *VMS Network Control Program Manual.*

3.5 Circuit Commands

The four classes of circuit that DECnet-VAX supports are DDCMP, CI, Ethernet, and X.25 circuits. Using NCP commands, you must identify all DECnet circuits connected to the local node and all permanent virtual circuits connected to local DTEs, and specify parameters that affect the operation of the circuits. The following sections describe circuit identification and discuss how to use NCP commands to specify circuit parameters.

3.5.1 Circuit Identification

Like nodes, circuits must also have unique identifiers. DECnet circuits and X.25 circuits are identified differently.

3.5.1.1 DDCMP Circuit Identification

For the VMS operating system, DDCMP circuit identification and line identification take one of the following formats:

dev-c

dev-c-u

dev-c.t

dev-c-u.t

where:

dev Is a device name. (Refer to the *VMS Network Control Program Manual* for a complete list of mnemonic device names.)

c Is a decimal number (0 or a positive integer) that designates the hardware controller for the device.

u Is a decimal unit or circuit number (0 or a positive integer) that is included only if there is more than one unit associated with the controller.

t Is a decimal number (0 or a positive integer) that identifies a tributary on a multipoint circuit. This is a logical tributary number, not to be confused with the tributary address that is used to poll the tributary.

Note: Circuit devices that are similar in operation are referred to by the same mnemonic.

DDCMP Point-to-Point Addressing

The following command specifies a synchronous point-to-point circuit. The command identifies the DMC (or DMR) circuit device and controller number 0.

```
NCP>SET CIRCUIT DMC-0 STATE ON
```

The following command specifies an asynchronous point-to-point circuit. The command identifies the DZ11 asynchronous circuit device by the mnemonic TT, and specifies controller number 0 and unit number 0 (that is, TTA0).

```
NCP>SET CIRCUIT TT-0-0 STATE ON
```

Dynamic asynchronous DDCMP circuit names are supplied automatically when you establish a dynamic connection. Note that you must load the asynchronous driver NODRIVER before establishing a dynamic connection.

The VMS operating system maps network management circuit names to system-specific circuit names (for example, DMC–4 maps to XME0). Network management circuit names provide network-wide circuit identification independent of individual operating system conventions.

DDCMP Multipoint Tributary Addressing

The following command identifies the DMP circuit device, controller number 0, and logical tributary 1:

```
NCP>SET CIRCUIT DMP-0.1 STATE ON
```

Use the SET CIRCUIT command to turn on the DMP circuit device as a multipoint tributary device.

DECnet–VAX software uses a form of circuit identification called a tributary address to poll a tributary for a specified circuit. Use the SET CIRCUIT command to establish the tributary address. For example, the following command specifies an address of 5 to tributary 1 on DMP controller 0:

```
NCP>SET CIRCUIT DMP-0.1 TRIBUTARY 5
```

Values from 1 to 255 are valid for this parameter. The node at the controlling end of this multipoint circuit uses this address to poll this line. You must set a corresponding tributary address on the remote node end of the circuit that will respond to a polling address of 5. For example:

```
NCP>SET CIRCUIT DMP-1.0 TRIBUTARY 5
```

The logical tributary number (0 in this case) is not to be confused with the tributary address. Refer to the description of logical tributary numbers in the circuit identification at the beginning of this section.

3.5.1.2 CI Circuit Identification

The TRIBUTARY parameter is also used to identify the CI node on the other end of a CI circuit. In the following example, the tributary address 1 identifies the CI node on the other end of circuit CI–0.1:

```
NCP>SET CIRCUIT CI-0.1 TRIBUTARY 1
```

The tributary node address is the CI port number of the remote CI node, not the DECnet node address.

Note that you must load the CNDRIVER before running DECnet over a CI (see Section 2.2.3).

3.5.1.3 Ethernet Circuit Identification

For the VMS operating system, Ethernet circuit identification takes the following format:

dev-c

where:

dev Is a device name.

c Is a decimal number (0 or a positive integer) that designates the hardware controller for the device.

For example, the following command identifies the circuit device UNA and the controller number 2 for an Ethernet circuit:

```
NCP>SET CIRCUIT UNA-2 STATE ON
```

3.5.1.4 X.25 Circuit Identification

Use the SET CIRCUIT command to identify X.25 circuits. The text following the X25– prefix in the command string identifies all PVCs and DLM SVCs. The text is an alphanumeric string not more than 12 characters in length. The entire string, including the prefix X25–, should not be longer than 16 characters. For example:

```
NCP>SET CIRCUIT X25-ANDIES ...
```

Specify unique circuit identifiers for each additional X.25 circuit. For example:

```
NCP>SET CIRCUIT X25-PVCONE ...
NCP>SET CIRCUIT X25-PVCTWO ...
      .
      .
      .
```

3.5.2 Circuit Parameters

The configuration database contains circuit parameters for all circuits connected to the local node or DTE. Table 3–2 lists the types of circuit and the circuit parameters that apply to each type. The circuit parameters supply information used to control various aspects of a circuit's operation. Table 3–3 lists the circuit parameters by function.

Table 3–2 Types of Circuit and Applicable Circuit Parameters

Type of Circuit	Applicable Circuit Parameter
All circuits	COUNTER TIMER seconds STATE { ON | SERVICE }
Circuits other than X.25 native PVCs	COST cost HELLO TIMER seconds
DDCMP circuits	ACTIVE BASE base ACTIVE INCREMENT increment BABBLE TIMER milliseconds DEAD THRESHOLD count DYING BASE base DYING INCREMENT increment DYING THRESHOLD count INACTIVE BASE base INACTIVE INCREMENT increment INACTIVE THRESHOLD count MAXIMUM BUFFERS count MAXIMUM TRANSMITS count POLLING STATE { ACTIVE | AUTOMATIC | DEAD | DYING | INACTIVE } SERVICE { ENABLED | DISABLED } TRANSMIT TIMER milliseconds TRIBUTARY tributary-address
DDCMP circuits and X.25 DLM circuits (PVCs or SVCs)	VERIFICATION { ENABLED | DISABLED | INBOUND }
Ethernet circuits	MAXIMUM ROUTERS number ROUTER PRIORITY number

Table 3–2 (Cont.) Types of Circuit and Applicable Circuit Parameters

Type of Circuit	Applicable Circuit Parameter
X.25 native PVCs	CHANNEL number DTE dte-address MAXIMUM DATA count MAXIMUM WINDOW count NETWORK network-name TYPE X25 USAGE PERMANENT
X.25 DLM circuits (PVCs or SVCs)	BLOCKING { ENABLED | DISABLED } CHANNEL number DTE dte-address NETWORK network-name MAXIMUM DATA count MAXIMUM RECALLS count MAXIMUM WINDOW count NUMBER dte-address OWNER EXECUTOR RECALL TIMER seconds TYPE X25 USAGE { INCOMING | OUTGOING | PERMANENT }

Table 3–3 Circuit Parameters and Their Functions

Parameter Function	Parameter
Indicates owner of circuit	OWNER EXECUTOR
Identifies circuit by address	TRIBUTARY tributary-address
Assigns circuit cost for routing purposes	COST number
Sets counter timer for circuit counter event logging	COUNTER TIMER seconds
Sets circuit's operational state	STATE { OFF | ON | SERVICE }

Table 3–3 (Cont.) Circuit Parameters and Their Functions

Parameter Function	Parameter
Controls DDCMP multipoint operation	ACTIVE BASE base DYING BASE base INACTIVE BASE base ACTIVE INCREMENT increment DYING INCREMENT increment INACTIVE INCREMENT increment DEAD THRESHOLD count DYING THRESHOLD count INACTIVE THRESHOLD count BABBLE TIMER milliseconds TRANSMIT TIMER milliseconds MAXIMUM BUFFERS count MAXIMUM TRANSMITS count POLLING STATE { ACTIVE | AUTOMATIC | DEAD | DYING | INACTIVE } TRIBUTARY
Sets timer to control Routing layer	HELLO TIMER seconds
Determines whether service operations are allowed for circuit (initiated locally or remotely)	SERVICE { ENABLED | DISABLED }
Controls Routing layer initialization of adjacent node	VERIFICATION { DISABLED | ENABLED | INBOUND }
Limits number of routers permitted on Ethernet circuit	MAXIMUM ROUTERS number
Sets priority of router on Ethernet circuit for selection of designated router	ROUTER PRIORITY number
Associates logical channel number with X.25 PVC	CHANNEL number
Specifies the network to which the local DTE belongs	NETWORK network-name
Associates local DTE with X.25 PVC or SVC	DTE dte-address
Assigns remote DTE address used to establish an outgoing DLM SVC or to reserve an incoming DLM SVC	NUMBER dte-address
Defines circuit type; if circuit is not X.25, DDCMP is assumed	TYPE X25
Controls data packet parameters for X.25 circuits	MAXIMUM DATA count MAXIMUM WINDOW count
Determines whether message blocking over DLM circuits occurs	BLOCKING { DISABLED | ENABLED }
Controls retransmission of DLM outgoing SVCs	MAXIMUM RECALLS count RECALL TIMER seconds

Use the SET CIRCUIT command to set and modify the parameters in Table 3–3. Use the CLEAR CIRCUIT command to reset them to their default values (if any) or to remove them from the volatile database. The circuit must be in the OFF state before you specify the ALL parameter in the CLEAR CIRCUIT command. The circuit must also be in the OFF state if you want to modify any parameters other than COST, COUNTER TIMER, SERVICE, STATE, and VERIFICATION.

Note that not all circuit devices support all parameters listed in Table 3–2 and Table 3–3. If a particular device does not support a parameter, an error message may be displayed. For information about specific circuit devices, refer to the *VMS I/O User's Reference Manual: Part II*.

3.5.2.1 Operational State of the Circuit

Just as you can control the operational state of the local node, you can also control the operational state of circuits connected to it. There are three circuit states:

OFF	Allows no traffic over a circuit. The circuit is unavailable for network activity.
ON	Allows traffic over the circuit. This is the normal operational state allowing for complete route-through and downline loading operations.
SERVICE	Restricts the circuit to service operations only. Only an Ethernet circuit allows logical link activity or route-through at the same time as service operations. Service operations include downline system loading, upline dumping, and loopback testing.

Use the STATE parameter to specify the operational state of a circuit. For example, the following command allows normal traffic over circuit DMC–0:

```
NCP>SET CIRCUIT DMC-0 STATE ON
```

DECnet–VAX may automatically change the state of a DDCMP circuit for certain functions. For example, assume that you have set a DDCMP circuit to ON. Later, someone performs a circuit-level loopback test on that circuit without first setting the circuit state to SERVICE. Network management software automatically turns the circuit to the appropriate internal state (or substate) for the test. If the circuit state were displayed at that point, it would register as ON-LOOPING. When the circuit is in this state, it is in use for an active circuit loop test. This test is termed active because it was initiated on the local node. The local node enters the passive loopback state (ON-REFLECTING) whenever a remote node initiates a loopback test with the local node. When the test finishes, the circuit returns to the ON state. For a complete list of circuit states, substates, and their transitions, refer to the *VMS Network Control Program Manual*.

Several circuit substates have the prefix AUTO. These substates can occur when an adjacent node is or is about to be in an automatic downline loading or triggering stage. For example, if circuit DMC–2 is in the ON state and the local node (BOSTON) receives a request for a downline load on that circuit, the network software on the local node automatically sets the circuit to the ON-AUTOSERVICE state.

Before performing service operations over a DDCMP circuit, you must enable that circuit. To do so, set the SERVICE parameter, which enables or disables service operations over a circuit. For example, the following command permits the circuit DMC–0 to be put in the SERVICE state, allowing service functions:

```
NCP>SET CIRCUIT DMC-0 SERVICE ENABLED
```

To disable a DDCMP circuit, set the SERVICE parameter to DISABLED, which allows you to restrict the operation of a circuit for network users. The default for the SERVICE parameter is DISABLED.

3.5.2.2 Circuit Timers

Two timers exist for controlling message transmissions and checking the status of adjacent nodes. The first is a hello timer, which defines the frequency of Routing layer Hello ("I'm still here") messages sent to the adjacent node on the circuit. The second is a listen timer, which controls the maximum amount of time allowed to elapse before the Routing layer stops waiting for either a Hello message or a user message from the adjacent node on the circuit. You cannot set the listen timer with an NCP command; the value of the listen timer is always twice the value of the hello timer at the local node.

To set the hello timer, enter the following command:

```
NCP>SET CIRCUIT DMP-0 HELLO TIMER 15
```

This command sets a limit of 15 seconds between Hello messages from the executor node to the adjacent node on circuit DMP–0. The listen interval is 30 seconds between messages from the node on circuit DMP–0 adjacent to the executor node. For the HELLO TIMER parameter, you must specify a value between 1 and 8191 seconds. The default value for the HELLO TIMER parameter is 15 seconds.

The value of the HELLO TIMER parameter should be the same on all adjacent nodes over the same circuit.

It is recommended that you accept the default value for the HELLO TIMER parameter, particularly if your node will communicate with nodes having versions of Network Management software lower than Version 3.0.

3.5.3 DDCMP Circuit Parameters

DDCMP circuit parameters include parameters related to verification and control of tributaries.

3.5.3.1 DDCMP Circuit Level Verification

The VERIFICATION parameter applies to DDCMP circuits and to X.25 DLM circuits (PVCs and SVCs).

The VERIFICATION parameter controls whether the local node checks the Routing layer passwords (RECEIVE PASSWORD and TRANSMIT PASSWORD) in the database entry for the remote node before it completes a node initialization request from that node.

To turn on verification, enter the following command:

```
NCP>SET CIRCUIT DMP-0 VERIFICATION ENABLED
```

This command specifies that the Routing layer will perform initialization of the remote node connected to circuit DMP–0. To turn verification off, enter the following command:

```
NCP>SET CIRCUIT DMP-0 VERIFICATION DISABLED
```

The default is DISABLED, which means that you need not specify a node in the configuration database to complete Routing layer initialization. To include a remote node in the configuration database, you must specify the NODE NAME and ADDRESS parameters; you can optionally specify the RECEIVE PASSWORD and TRANSMIT PASSWORD parameters.

When a remote node submits a node initialization request to the local node, the following rules apply:

- Nodes not defined in the remote node database at the local node cannot initialize over a circuit that has verification enabled.
- Nodes defined in the remote node database for which receive and transmit passwords are not specified are allowed to initialize whether or not verification is enabled on the circuit.
- Nodes defined in the remote node database for which receive and transmit passwords are specified are allowed to initialize over a circuit with verification enabled, provided the receive password in the local database matches the transmit password sent by the remote node.
- Any node is allowed to initialize over a circuit for which verification is disabled.

The VERIFICATION INBOUND parameter applies to any DDCMP point-to-point circuit. When you specify VERIFICATION INBOUND, the remote node submitting an initialization request to the local node must supply a transmit password that matches the receive password for that node in the local node database. The local node, however, does not send its initialization password to the requesting node. The VERIFICATION INBOUND parameter provides added security for the local node, which can verify the password of a node requesting a connection without revealing its own password.

For example, to require that a remote node supply a password before it can initialize on circuit DMP-0 when the local node does not supply a password, enter the following command:

```
NCP>SET CIRCUIT DMF-0 VERIFICATION INBOUND
```

The VERIFICATION INBOUND parameter is supplied automatically for a dynamic asynchronous DDCMP circuit. When a dialup node requests a dynamic connection to the local node and the VERIFICATION INBOUND parameter is set for the circuit, you must specify the INBOUND parameter for the dialup node in the node database. If you do not specify VERIFICATION INBOUND, the INBOUND parameter in the dialup node entry is ignored.

3.5.3.2 DDCMP Tributary Control

Several circuit parameters enable you to regulate and control tributaries. Some of these parameters apply to polling, others to timers. Note that you specify these circuit parameters on the control station, not on the tributary itself.

Polling Over DDCMP Circuits

To control the polling state of a tributary, use the DYING THRESHOLD, DEAD THRESHOLD, or INACTIVE THRESHOLD parameters. There are four polling states: ACTIVE, INACTIVE, DYING, and DEAD. These parameters determine the number of times the control station polls the active, inactive, or dying tributary before changing its polling state. For example, the following command sets the polling threshold for circuit DMP-0.3:

```
NCP>SET CIRCUIT DMP-0.3 DYING THRESHOLD 5
```

The control station attempts to poll its tributary 5 times. If it gets receive timeouts for five consecutive polls, the control station changes the tributary's polling state from ACTIVE or INACTIVE to DYING. Values for the DYING THRESHOLD parameter range from 0 to 255 and the default is 2. The following command sets the polling threshold for circuit DMP–0.1:

```
NCP>SET CIRCUIT DMP-0.1 INACTIVE THRESHOLD 12
```

The control station attempts to poll its active tributary 12 times. If it receives only acknowledgments, but no data responses, the control station changes the active tributary's polling state to INACTIVE. The values for the INACTIVE THRESHOLD parameter range from 0 to 255 and the default is 8.

You can lock a tributary into one of the four states by using the POLLING STATE parameter. Usually, the tributary's state is allowed to vary according to the multipoint polling algorithm. This variance occurs when this parameter is set to AUTOMATIC. Use this parameter to lock a tributary into the ACTIVE, INACTIVE, DYING, or DEAD state. For example, the following command locks the tributary controlled by circuit DMP–0.1 into a DEAD state:

```
NCP>SET CIRCUIT DMP-0.1 POLLING STATE DEAD
```

The base priority of a tributary is the lowest value to which that tributary can be set after a poll. A control station polls tributaries with high priorities first. Note that a control station does not poll tributaries with priorities below 128. To specify the base priority for a tributary, use the ACTIVE BASE, INACTIVE BASE, or DYING BASE parameters. After polling the tributary, the control station resets the base priority of the active, inactive, or dying tributary to this value. You can set a separate base value for each of the polling states, as shown in the following example:

```
NCP>SET CIRCUIT DMP-1.2 ACTIVE BASE 225
```

After a poll, this command resets the base priority of the tributary on circuit DMP–1.2 to 225. The values for all BASE parameters range from 0 to 255. The defaults are ACTIVE, 255; INACTIVE, 0; and DYING, 0.

You can also increment the priority of a tributary each time the line-scheduling timer expires. If, for instance, the polls pass over a tributary with a low priority, you can raise the priority of that tributary by using the ACTIVE INCREMENT, INACTIVE INCREMENT, or DYING INCREMENT parameter. When the scheduling timer expires on an unpolled tributary, it increases the priority according to the value you set. You can set a separate increment value for each polling state, as shown in the following example:

```
NCP>SET CIRCUIT DMP-2.2 INACTIVE INCREMENT 200
```

This command adds 200 to the base priority of the tributary on circuit DMP–2.2. The increment values range from 0 to 255. The defaults are ACTIVE, 0; INACTIVE, 64; and DYING, 16. Note that, if you set a 0 increment on a tributary with a base priority lower than 128, the tributary will never be polled. Active tributaries usually have a high base priority and, therefore, do not need a high increment value.

The MAXIMUM BUFFERS and MAXIMUM TRANSMITS parameters give you further control over the tributary. MAXIMUM BUFFERS specifies the maximum number of buffers that a tributary can use from the common buffer pool. If you do not set this parameter explicitly, the default is 4. Values for this parameter can be either integers ranging from 1 to 254 or the word

UNLIMITED. For example, the following command sets an upper limit of 10 buffers that the tributary on this circuit can use from the common buffer pool:

```
NCP>SET CIRCUIT DMP-0.2 MAXIMUM BUFFERS 10
```

The MAXIMUM TRANSMITS parameter specifies the maximum number of data messages that the tributary can transmit in a single poll interval. Values range from 1 to 255; the default is 4. For example, the following command sets an upper limit of two data message transmits from the tributary on circuit DMP-0.1:

```
NCP>SET CIRCUIT DMP-0.1 MAXIMUM TRANSMITS 2
```

DDCMP Tributary Circuit Timers

Two timers exist for controlling message retransmission at the DDCMP tributary circuit level. The babble timer controls the amount of time that a tributary or remote half-duplex station can transmit; the transmit timer sets the amount of time to delay between data message transmissions. To specify these timers, enter the following commands:

```
NCP>SET CIRCUIT DMP-0.1 BABBLE TIMER 8000

NCP>SET CIRCUIT DMP-0.1 TRANSMIT TIMER 4000
```

The first command limits transmission time to 8 seconds (8000 milliseconds) for the circuit's tributary. Values for the BABBLE TIMER parameter range from 1 to 65,535; the default is 6000 (6 seconds).

The second command sets a delay of 4 seconds (4000 milliseconds) between each transmission from the tributary. Values for the TRANSMIT TIMER parameter range from 0 to 65,535; the default is 0.

3.5.4 Ethernet Circuit Parameters

Parameters that Ethernet circuits have in common with other DECnet–VAX circuits are HELLO TIMER, COST, COUNTER TIMER and STATE. Parameters unique to Ethernet circuits are ROUTER PRIORITY and MAXIMUM ROUTERS, which you can specify in the SET CIRCUIT command.

If there are two or more routers on the same Ethernet, the one with the highest numerical priority (up to a maximum value of 127) is elected the designated router. The designated router provides message routing services for end nodes on the Ethernet (see Section 2.4.4.1). A designated router is selected even if there are currently no end nodes on the Ethernet. Note that routers are not required in order to route messages over the Ethernet on behalf of end nodes; Ethernet end nodes are capable of communicating directly. However, routers are required to route messages off of the Ethernet over other circuits such as DDCMP circuits.

Use the SET CIRCUIT command to set the ROUTER PRIORITY value in the applicable circuit database at the executor node. For example, the following command assigns a router priority of 70 to the local node on circuit UNA–0:

```
NCP>SET CIRCUIT UNA-0 ROUTER PRIORITY 70
```

Each node on Ethernet circuit UNA-0 is assigned a router priority value in the range 0 through 127; the default value is 64. DECnet software compares the router priority values of the nodes and elects the router with the highest priority the designated router. If two or more nodes on the Ethernet have the same highest router priority value, the node with the highest node address is selected as designated router. To learn which router is the designated router, enter a SHOW ACTIVE CIRCUITS CHARACTERISTICS command. The following information is displayed for circuit UNA-0:

```
Designated router          = 1.224 (ROBIN)
Router priority            = 70
```

The recommended limit on the number of routers on an Ethernet circuit is 10, because of the control traffic overhead (composed of routing messages and Hello messages) involved. The maximum number of routers allowed is 33. The MAXIMUM ROUTERS parameter specifies the maximum number of routers (other than the executor node) that the Routing layer is to allow on a particular Ethernet circuit. Use the SET CIRCUIT command to assign the MAXIMUM ROUTERS value for an Ethernet circuit. For example, the following command sets a maximum value of 4 to the number of routers (in addition to the executor node) that are permitted on circuit UNA-0:

```
NCP>SET CIRCUIT UNA-0 MAXIMUM ROUTERS 4
```

The default value is 33.

3.5.5 Ethernet Configurator Module Commands

Use the Ethernet configurator module to obtain a list of all systems on an Ethernet circuit or circuits. Each DIGITAL-supported node on an Ethernet circuit periodically transmits a system identification message to a multicast address to which the Ethernet configurator listens. The configurator uses these messages to build the configuration list.

Use NCP commands to access and control the configurator module. The configurator runs as a separate process, available to all users on the system. After the configurator starts, it continues to execute, maintaining and updating its database of information on active nodes until a user causes it to stop listening to system identification messages.

If several users of a particular system enter SET MODULE CONFIGURATOR commands, they all access the same configurator module. To determine whether the configurator module is already running, enter the following command:

```
NCP>SHOW MODULE CONFIGURATOR KNOWN CIRCUITS
```

3.5.5.1 Enabling Surveillance by the Ethernet Configurator

To create or modify Ethernet configurator module parameters in the volatile database, use the SET MODULE CONFIGURATOR command. The SURVEILLANCE ENABLED parameter in this command causes the configurator module to begin listening to system identification messages transmitted by all systems on the circuit or circuits specified in the command. The configurator collects this information and constructs a list of systems and their capabilities in the volatile database.

3.5.5.2 Obtaining a List of Systems on Ethernet Circuits

To obtain information about the current configuration of nodes on Ethernet circuits, use the SHOW MODULE CONFIGURATOR command. This command permits you to access the configurator volatile database, which contains the following information for each system:

- The Ethernet physical and hardware addresses of the system
- The device connecting the system to the Ethernet
- The maintenance version number of the system
- A list of maintenance functions that the node can perform
- The last time a system identification message was received from that system

The SHOW MODULE CONFIGURATOR command causes the configurator to display this information along with the amount of time surveillance has been enabled on the circuit. For example:

```
NCP>SHOW MODULE CONFIGURATOR CIRCUIT UNA-0 STATUS
```

For circuit UNA–0, this command results in the following display:

```
Module Configurator Volatile Status as of 30-DEC-1988 09:15:25

Circuit name              = UNA-0
Surveillance flag         = enabled
Elapsed time              = 00:32:43
Physical address          = AA-00-04-00-A3-04
Time of last report       = 30-DEC 9:14:8
Maintenance version       = V3.0.0
Function list             = Loop, Primary loader
Hardware address          = AA-00-03-00-00-07
Device type               = UNA

Circuit name              = UNA-0
Surveillance flag         = enabled
Elapsed time              = 0:32:43
Physical address          = AA-00-04-00-A1-04
Time of last report       = 30-DEC 9:11:29
Maintenance version       = V3.0.0
Function list             = Loop, Primary loader
Hardware address          = AA-00-03-00-00-57
Device type               = UNA
```

3.5.5.3 Disabling Surveillance by the Ethernet Configurator

To cause the configurator to stop listening to system identification messages on specific Ethernet circuits, use the SURVEILLANCE DISABLED parameter in the SET MODULE CONFIGURATOR command. If you specify the KNOWN CIRCUITS parameter with this command, the configurator no longer listens to system identification messages being broadcast on any Ethernet circuit known to the local node. For example, the following command causes the configurator to cease surveillance of all Ethernet circuits known to the local node:

```
NCP>SET MODULE CONFIGURATOR KNOWN CIRCUITS -
_ SURVEILLANCE DISABLED
```

After the configurator ceases surveillance of all Ethernet circuits it has been monitoring, the list of system information is deleted.

3.5.6 X.25 PVC Parameters

The circuit parameters described in the following sections apply to permanent virtual circuits (PVCs) used for X.25 native operations or for DLM. In addition, for DLM PVCs, the parameters described in Section 3.5.7 also apply.

3.5.6.1 Parameters Common to X.25 Circuits

The TYPE X25 and USAGE parameters are common to all X.25 circuits.

Use the TYPE parameter to specify an X.25 circuit, as follows:

```
NCP>SET CIRCUIT X25-ANDIES ... TYPE X25 ...
```

Note that, by default, when the name of the circuit starts with "X25," for example, X25–ANDIES, the circuit type is X.25. The TYPE parameter is optional.

Use the USAGE parameter to specify that the circuit is a PVC, as follows:

```
NCP>SET CIRCUIT X25-ANDIES ... USAGE PERMANENT ...
```

USAGE PERMANENT indicates that the circuit is permanently connected to a remote DTE and does not need to be switched dynamically. The USAGE parameter is mandatory for PVCs and takes no default.

3.5.6.2 Permanent Virtual Circuit Parameters

When PVCs are first specified, the CHANNEL and DTE parameters are mandatory. In addition, the NETWORK parameter is mandatory if more than one network is set up.

Use the CHANNEL parameter to associate a logical channel number with each PVC. This number is allocated to you by the PSDN at subscription time and is in the range 1 to 4095. Each PVC must have a unique channel number different from those previously specified for outgoing calls in the SET MODULE X25–PROTOCOL command. The following command illustrates the use of this CHANNEL parameter:

```
NCP>SET CIRCUIT X25-ANDIES ... CHANNEL 3 ...
```

Use the DTE parameter to associate the local DTE address with each PVC. The following command illustrates the use of the DTE parameter:

```
NCP>SET CIRCUIT X25-ANDIES ... DTE 123789456 ...
```

The DTE address is a decimal integer of 1 to 15 digits and must have been specified previously in a SET MODULE X25–PROTOCOL command.

The NETWORK parameter defines the network to which the DTE connects. For example:

```
NCP>SET CIRCUIT X25-ANDIES ...NETWORK TELENET1
```

The network must have been defined previously with a SET MODULE X25–PROTOCOL command.

3.5.6.3 Data Packet Control

Two parameters control the transmission of data packets over the PVC: MAXIMUM DATA and MAXIMUM WINDOW.

The MAXIMUM DATA parameter specifies the maximum size of the packet for a particular circuit. For example, the following command sets the maximum size of the packet to 128 bytes for the circuit ANDIES:

```
NCP>SET CIRCUIT X25-ANDIES ... MAXIMUM DATA 128 ...
```

The maximum packet size must always be at least 5 bytes smaller than the maximum size of the frame on a line (see Section 3.6.5.1). Specify a value that is a power of 2 in the range 16 to 4096 bytes.

The MAXIMUM DATA parameter is optional and, by default, takes the value specified for the local DTE. See the *Public Network Information* manual for the default value of this parameter.

The MAXIMUM WINDOW parameter specifies the window size for a particular PVC. For example, the command that follows sets the window size to 2 for the circuit X25–ANDIES:

```
NCP>SET CIRCUIT X25-ANDIES ... MAXIMUM WINDOW 2 ...
```

Specify a value in the range 1 to 127.

The MAXIMUM WINDOW parameter is optional and, by default, takes the value specified for the local DTE. See the *Public Network Information* manual for the default value of this parameter.

Both parameters, if specified, must be set to values that agree with your PSDN subscription.

3.5.7 DLM Circuit Parameters

A data link mapping (DLM) circuit allows an X.25 virtual circuit to be used as a DECnet data link in communicating with other DECnet nodes over a PSDN. Several circuit parameters are specific to the operation of DLM circuits: the OWNER EXECUTOR parameter; the remote DTE address and the network name used by DECnet to establish a DLM outgoing switched virtual circuit; the USAGE parameter for a DLM circuit; and two parameters to regulate recalls for DLM outgoing SVCs.

Parameters that DLM circuits have in common with other X.25 circuits are CHANNEL, DTE, MAXIMUM DATA, and MAXIMUM WINDOW. The circuit parameter VERIFICATION applies to DLM circuits and DDCMP circuits.

3.5.7.1 DLM Circuit Owner

Use the OWNER EXECUTOR parameter to indicate that the Routing layer has exclusive rights to use the circuit. To specify that the circuit X25–DLMOUT should be used as a DLM circuit, enter the following command:

```
NCP>SET CIRCUIT X25-DLMOUT OWNER EXECUTOR
```

The OWNER EXECUTOR parameter is required for a DLM circuit.

3.5.7.2 Remote DTE Addresses

To establish an SVC with a remote DTE, DECnet software requires the address of the remote DTE. Use the NUMBER parameter in the SET CIRCUIT command to specify the remote DTE address for incoming or outgoing DLM SVCs.

For outgoing calls, the Routing layer uses this address (and the subaddresses required at the remote DTE) to call on the circuit. For example:

```
NCP>SET CIRCUIT X25-DLMOUT NUMBER 31191234567842
```

Outgoing calls on circuit X25–DLMOUT use the DTE 311912345678 and a subaddress of 42 to establish an SVC with a remote DTE associated with this address.

For incoming calls, use the NUMBER parameter to force them to a particular circuit on the basis of the remote DTE address. If you specify a NUMBER parameter for each incoming DLM circuit at the local DTE, an incoming call from a remote DTE is rejected if its address does not match the number specified for any incoming circuit. If any incoming circuit does not have a number specified, then the circuit can handle calls from any DTE. You can also use the NUMBER parameter with an incoming circuit to specify additional routing parameters for a selected DTE. For example:

```
NCP>SET CIRCUIT X25-INC USAGE INCOMING -
_ NUMBER 31191234567842 COST 15
```

Circuit X25–INC receives calls only from the remote DTE with the DTE 311912345678 and subaddress 42. In this example, a cost of 15 is assigned to the DLM connection to remote DTE 311912345678 to reflect a higher routing cost for this configuration.

Note: For an outgoing SVC, if there is more than one network available, you must specify which network to use for the outgoing call.

3.5.7.3 Recalls for DLM Circuits

If previous attempts to establish a DLM SVC have been unsuccessful, DECnet–VAX attempts to recall a number. You can set the frequency of recalls and the maximum number of recalls DECnet attempts by using two parameters. The RECALL TIMER parameter sets the interval that DECnet should wait before attempting to place a call to establish an SVC. The MAXIMUM RECALLS parameter specifies the maximum number of times DECnet should attempt to place a call to establish an SVC. The following command causes DECnet–VAX to place a call every 10 seconds for a maximum of 10 times to establish an SVC for circuit X25–DLMOUT:

```
NCP>SET CIRCUIT X25-DLMOUT RECALL TIMER 10 -
_ MAXIMUM RECALLS 10
```

The default value for the RECALL TIMER parameter is 100 seconds. The range of acceptable values is 1 to 255. If an attempt to make an outgoing call causes the system to exceed the MAXIMUM RECALLS parameter, the circuit is placed in the ON-FAILED state, and you must enter the following command before the outgoing call can be attempted again:

```
NCP>SET CIRCUIT X25-DLMOUT STATE ON
```

3.5.7.4 DLM Circuit Usage

DLM circuits operate according to the usage you define for them in the volatile database. You may use DLM SVCs either for outgoing or incoming calls. The usage of DLM PVCs is PERMANENT; that is, the circuit is permanently connected to a remote DTE, and does not need to be switched dynamically. The USAGE parameter specifies how DLM circuits are to be used, as in the following example:

```
NCP>SET CIRCUIT X25-DLMOUT USAGE OUTGOING ...
```

3.5.7.5 Executor Node Subaddresses

When you are configuring the network, you can optionally define a range of subaddresses that the DECnet Routing layer accepts as incoming DLM calls. VAX PSI routes all incoming calls within the specified subaddress range to the Routing layer, to be handled as DLM circuits. You are responsible for ensuring that the subaddress specified in the outgoing DLM NUMBER parameter (see Section 3.5.7.2) matches the range of subaddresses on the incoming side, as specified in the EXECUTOR SUBADDRESSES parameter of the SET EXECUTOR command.

When the Routing layer receives an incoming call from a DTE, it scans the incoming DLM circuits to find the address of the DTE sending the call that matches the remote DTE address specified for the circuit (in the NUMBER parameter of the SET CIRCUIT command). If an incoming circuit does not have the NUMBER parameter specified, that circuit is selected to accept any incoming call that has not yet been matched to a particular circuit. If all incoming circuits have the NUMBER parameter specified and a call is received from a DTE whose address does not match any circuit, that call is rejected.

Use the SUBADDRESSES parameter in the SET EXECUTOR command to specify executor subaddresses (Section 3.3.1 describes the SET EXECUTOR command). For example, use the SET EXECUTOR command to modify subaddresses in the volatile database:

```
NCP>SET EXECUTOR SUBADDRESSES 42
```

This command indicates that the Routing layer is to handle only incoming X.25 calls that specify local DTE subaddress 42. All other calls are handled by VAX PSI. A subaddress may consist of a range. For example, the following command indicates that the Routing layer handles all incoming X.25 calls that specify a local DTE subaddress in the range of 42 to 50:

```
NCP>SET EXECUTOR SUBADDRESSES 42-50
```

When specifying a subaddress range, use an integer in the range of 0 to 9999. Separate two subaddresses with a hyphen to indicate a range, and be sure the second subaddress is always greater than the first.

3.5.7.6 Setting Up a DLM Circuit

The following example illustrates how to set up a DLM circuit between DECnet node A (DTE address 123) and DECnet node B (DTE address 456), which are to communicate over a PSDN.

On node A, enter the following command to specify the outgoing DLM circuit in the volatile database:

```
NCP>SET CIRCUIT X25-OUTGOING -
_ TYPE X25 -
_ OWNER EXECUTOR -
_ USAGE OUTGOING -
_ NUMBER 4561 -
_ STATE ON
```

The command must specify the DTE address and subaddress of the remote node. The value in the NUMBER parameter represents DTE 456, subaddress 1.

On node B, the following command indicates that the DECnet Routing layer is to accept all incoming calls to this node (DTE) that have a subaddress in the range 1 to 20:

```
NCP>SET EXECUTOR SUBADDRESSES 1-20
```

Note that you must enter this command at node B before node B can accept calls from node A.

You then enter the following command at node B to specify the incoming DLM circuit in the volatile database:

```
NCP>SET CIRCUIT X25-INCOMING -
_ TYPE X25 -
_ OWNER EXECUTOR -
_ USAGE INCOMING -
_ STATE ON
```

3.5.8 Circuit Counters

DECnet–VAX automatically maintains certain statistics for circuits in the network. These statistics are known as circuit counters. For all circuits, counter information may include the number of data packets sent, received, and lost over the circuit; timeouts; and the amount of time since the counters were last zeroed. For DDCMP circuits, counters are maintained for timeouts and data and buffer errors, and, for both DDCMP and Ethernet circuits, the number of bytes and data blocks sent and received. For X.25 circuits, the following statistics are indicated in counters: the time since the counters were zeroed; the number of bytes, data blocks, and resets sent and received; and the number of resets initiated by the network. Information obtained from counters may be useful either alone or in conjunction with logging information to measure the performance and throughput for a given circuit. See the *VMS Network Control Program Manual* for a complete list of circuit counters. Refer to Section 2.9 for a discussion of logging.

You can use NCP to regulate the frequency with which circuit counters are logged and when they are zeroed. At any point while the network is running, you can also display circuit counter statistics using the SHOW CIRCUIT COUNTERS command.

To set a timer whose expiration automatically causes the circuit counters to be logged at the logging sink and then zeroed, use the SET CIRCUIT command with the COUNTER TIMER parameter. The following command causes a circuit counter logging event to take place every 600 seconds:

```
NCP>SET CIRCUIT DMC-0 COUNTER TIMER 600
```

To clear this parameter, enter the following NCP command:

```
NCP>CLEAR CIRCUIT DMC-0 COUNTER TIMER
```

At any point when the network is running, you can zero counters for a given circuit or for all known circuits. Enter the following commands to zero circuit counters:

```
NCP>ZERO CIRCUIT DMC-0 COUNTERS
NCP>ZERO KNOWN CIRCUITS COUNTERS
```

3.6 Line Commands

DECnet–VAX supports four classes of line: DDCMP, CI, Ethernet, and X.25. You must use NCP commands to identify all physical lines connected to the local node and to specify parameters that affect operation of the lines. The following sections describe line identification and discuss the line parameters you can use.

3.6.1 Line Identification

As with nodes and circuits, lines must have unique identifiers. The line and circuit names identify a logical connection. For the VMS operating system, line identification takes one of the following formats:

dev-c

dev-c[-u]

where:

dev Is the device name. (Refer to the *VMS Network Control Program Manual* for a complete list of mnemonic device names.)

c Is the decimal number (0 or a positive integer) designating the device's hardware controller.

u Is the decimal unit or line number (0 or a positive integer) included if the device is a multiple unit line controller. For all non-multiplexed lines, the unit number is optional and, if specified, is always zero (0).

Note: Devices that are similar in operation are referred to by the same mnemonic.

The VMS operating system maps network management line names to system-specific line names (for example, DMC–4 maps to XME0). Network management line names provide network-wide line identification independent of individual operating system conventions.

Commands in this section illustrate line identification.

The following command specifies a synchronous DDCMP point-to-point line, identifying the DMC (or DMR) line device and controller number 0:

```
NCP>SET LINE DMC-0 STATE ON
```

The following command specifies an asynchronous DDCMP point-to-point line. It identifies the DMF32 asynchronous line unit by the mnemonic TX and specifies controller number 0 and unit number 0 (that is, TXA0).

```
NCP>SET LINE TX-0-0 RECEIVE BUFFERS 4 STATE ON
```

When you turn on an asynchronous line, you are advised to set the number of receive buffers to a value of 4 or more (see Section 3.6.3.1).

Note that dynamic asynchronous DDCMP line names are supplied automatically when a dynamic connection is established.

The following command specifies the CI line CI-0:

```
NCP>SET LINE CI-0 STATE ON
```

The following command specifies the Ethernet line UNA-0:

```
NCP>SET LINE UNA-0 STATE ON
```

The following command specifies the X.25 line DUP-0:

```
NCP>SET LINE DUP-0 STATE ON
```

For each X.25 line, specify a unique device, for example:

```
DMF-0
DUP-0
KMX-0-0
KMX-0-1
KMV-0
```

3.6.1.1 Line Protocols

As part of the process of identifying lines, you must specify the line protocol. To ensure that the data link protocol operates properly when information is transferred over a line, use the SET LINE command with the PROTOCOL parameter to specify a line protocol. The protocols are as follows:

DDCMP CONTROL	Specifies the line as a multipoint control station. You can set multiple circuits for CONTROL lines. Each circuit must have a unique physical tributary address.
DDCMP DMC	Specifies that the line is in DMC emulator mode. DMC is similar to DDCMP POINT protocol, except that DMC uses an older version of DDCMP (Version 3.2). This protocol should be set for the local line when the remote line is a DMC.
DDCMP POINT	Specifies the line as one end of a point-to-point DDCMP connection. You may specify only one circuit per POINT line.
DDCMP TRIBUTARY	Specifies that the line is a multipoint tributary end of a DDCMP multipoint group. You may specify only one circuit per TRIBUTARY line.
ETHERNET	Specifies that the line is a multiaccess line that uses the Ethernet protocol.

LAPB	Specifies that the line uses the X.25 level 2 protocol. The line must be used by the X25–PROTOCOL module.
LAPBE	Specifies that the line uses the X.25 level 2 protocol with extended sequencing. The line must be used by the X25–PROTOCOL module.

Note that you do not specify any protocol for a CI line. The CI uses its own private protocols for communication between nodes.

If you do not specify a line protocol, the following default values apply, according to the device specified.

Device	Default Protocol
BNA	ETHERNET
CI	None (not specified)
DMB	DDCMP POINT
DMC/DMR	DDCMP POINT
DMP/DMV	DDCMP POINT
DMF	DDCMP POINT
TT/TX	DDCMP POINT
DUP/DPV	LAPB
KMX	LAPB
KMV	LAPB
KMY	LAPB
QNA	ETHERNET
SVA	ETHERNET
UNA	ETHERNET

The SET LINE PROTOCOL examples that follow specify line protocols in the configuration database at the local node and on remote nodes other than DECnet–VAX, such as DECnet–RSX. For example, the following command identifies line DMP–0 as a multipoint control station:

```
NCP>SET LINE DMP-0 PROTOCOL DDCMP CONTROL
```

You set this parameter in the database of the local node at the controlling end of this line. You could specify a tributary for this line, as follows:

```
NCP>SET LINE DMP-1 PROTOCOL DDCMP TRIBUTARY
```

You set this parameter in the database of the remote node connected to the tributary end of the control station for that line.

3.6.2 Line Parameters

The configuration database should contain line parameters for all physical lines connected to the local node or DTE. These parameters supply information used to control various aspects of a line's operation. Table 3–4 lists the types of line and the line parameters applicable to them. Table 3–5 lists all line parameters by function.

Table 3–4 Types of Line and Applicable Line Parameters

Type of Line	Applicable Line Parameter
All lines	CONTROLLER controller-mode COUNTER TIMER seconds STATE { ON | OFF }
All lines except CI	PROTOCOL protocol-name
All lines except X.25 lines	BUFFER SIZE number
All lines except Ethernet lines	RETRANSMIT TIMER millisecond
DDCMP lines	CLOCK { EXTERNAL | INTERNAL } DEAD TIMER milliseconds DELAY TIMER milliseconds DUPLEX duplex-mode RECEIVE BUFFERS count SCHEDULING TIMER milliseconds SERVICE TIMER milliseconds STREAM TIMER milliseconds
DMR11 lines	TRANSMIT PIPELINE count
Asynchronous DDCMP lines	HANGUP { DISABLED | ENABLED } LINE SPEED number SWITCH option
X.25 lines	HOLDBACK TIMER milliseconds INTERFACE { DCE | DTE } MAXIMUM BLOCK count MAXIMUM RETRANSMITS count MAXIMUM WINDOW count MICROCODE DUMP file-spec NETWORK network-name PROTOCOL { LAPB | LAPBE } STATE SERVICE

Table 3–5 Line Parameters and Their Functions

Parameter Function	Parameter
Defines line protocol	PROTOCOL { DDCMP CONTROL | DDCMP POINT | DDCMP DMC | DDCMP TRIBUTARY | ETHERNET | LAPB | LAPBE }
Sets counter timer for line counter event logging	COUNTER TIMER seconds
Selects clock type	CLOCK { INTERNAL | EXTERNAL }
Sets line's operational state	STATE { OFF | ON | SERVICE }
Sets maximum receive buffer size for logical links over specific line	BUFFER SIZE number
Sets number of buffers in receive queue	RECEIVE BUFFERS number
Establishes physical line control parameters for DDCMP protocol	DUPLEX { FULL | HALF } DEAD TIMER milliseconds DELAY TIMER milliseconds RETRANSMIT TIMER milliseconds SCHEDULING TIMER milliseconds SERVICE TIMER milliseconds STREAM TIMER milliseconds
Specifies asynchronous DDCMP line characteristics	HANGUP { DISABLED | ENABLED } LINE SPEED number SWITCH { DISABLED | ENABLED }
Establishes line-level loopback control for controller operation	CONTROLLER { LOOPBACK | NORMAL }
Establishes frame control parameters for X.25 line	MAXIMUM BLOCK count MAXIMUM WINDOW count
Controls retransmission of frames for X.25 line	MAXIMUM RETRANSMITS count RETRANSMIT TIMER
Controls acknowledgment of frames for X.25 line	HOLDBACK TIMER milliseconds
Controls packet transmission over satellite link	TRANSMIT PIPELINE count
Defines the way in which the line is used	INTERFACE { DTE | DCE }

Use the SET LINE command to establish and modify the parameters in Table 3–4 and Table 3–5. You must set the line to OFF if you want to modify any parameters except COUNTER TIMER, SERVICE, SERVICE TIMER, and STATE. STATE is a required parameter for all lines that you specify in the configuration database. Use the CLEAR LINE command to reset parameters

to their default values (if any) in the volatile database. The line must be off before you specify the ALL parameter in the CLEAR LINE command.

Note that not all circuit devices support all parameters listed in Table 3–4 and Table 3–5. If a particular device does not support a parameter, an error message may be displayed. For information about specific circuit devices, refer to the *VMS I/O User's Reference Manual: Part II*.

3.6.2.1 Operational State of Lines

As with local node and circuit states, you can control the operational state of lines connected to the local node or to the local DTE. There are three possible line states:

OFF	Allows no traffic over a line. The line is unavailable for network activity.
ON	Allows traffic over the line. The ON state is the normal operational state, which allows complete route-through and downline loading operations.
SERVICE	Allows only restricted line service over the line. This traffic includes loopback testing (used only for X.25 lines).

The ON and SERVICE states have substates; see the *VMS Network Control Program Manual* for a complete list of line states, substates, and their transitions.

Use the STATE parameter to specify the operational state of a line. For example, to allow normal traffic over line DMC–0, enter the following command:

```
NCP>SET LINE DMC-0 STATE ON
```

The following command specifies the operational state of an X.25 line, allowing normal traffic over the DUP–0:

```
NCP>SET LINE DUP-0 STATE ON
```

The STATE parameter is optional and, by default, is set to OFF.

3.6.2.2 Buffer Size

You can increase the maximum size of receive buffers (and therefore the size of NSP messages) that can be transmitted over a particular line by specifying the BUFFER SIZE parameter in the SET LINE command. For certain logical links established over the line to adjacent nodes, this BUFFER SIZE value overrides the executor node BUFFER SIZE limit specified in the SET EXECUTOR command (see Section 3.3.5.1).

If you specify the BUFFER SIZE parameter for a line, the adjacent node on any new logical link initiated over that line can optionally accept an NSP message segment size that is based on the BUFFER SIZE value. If the remote node accepts the segment size, the logical link to that node is then tied to that circuit. If the circuit fails, the logical link does not automatically route the packet through an alternate circuit; that is, the logical link becomes nonadaptive.

For example, the following command sets the maximum size of receive buffers for line UNA–0 to 1400 bytes, but only for logical links to adjacent nodes that accept 1400 bytes as the NSP segment size:

```
NCP>SET LINE UNA-0 BUFFER SIZE 1400
```

If the adjacent node does not accept a segment size based on the BUFFER SIZE value, the default for any line except an Ethernet line is the executor node's BUFFER SIZE value. The default for an Ethernet line is 1498 bytes.

You can use this feature to maximize performance over high-speed links, such as Ethernet, by using a large value for the BUFFER SIZE parameter and causing all logical links between adjacent nodes on the Ethernet to use that larger message size.

3.6.3 DDCMP Line Parameters

Several parameters regulate various aspects of a DDCMP line's physical protocol operation. You can specify the number of receive buffers, the duplex mode, and the timers for both normal and service operations.

Parameters that apply specifically to asynchronous DDCMP lines indicate the speed of the line, whether modem signals are dropped when a line is shut down, and whether an asynchronous line is switched back to a terminal line when disconnected from the network. For a dynamic asynchronous line, DYNSWITCH supplies these parameters to NETACP automatically.

The following sections describe the DDCMP line parameters.

3.6.3.1 Line Buffers

To allocate buffers for data reception by the device driver for a particular DDCMP line, use the RECEIVE BUFFERS parameter. The following command sets four buffers for this line:

```
NCP>SET LINE DMC-1 RECEIVE BUFFERS 4
```

Values for this parameter range from 1 to 32. The number of buffers you set depends on throughput requirements and available memory pool. A value in the range of 2 to 4 is adequate for line speeds of less than 56K bits. For asynchronous lines, a value of at least 4 is recommended. Megabit line speeds may require 8 or more buffers, depending on the observed errors. For LAPB lines, see the description of X.25 line parameters in Section 3.6.5.

3.6.3.2 Duplex Mode

To set the duplex mode for a DDCMP line, use the DUPLEX parameter. For example, the following command sets the mode of the DMC11 device controller to full duplex for line DMC-1:

```
NCP>SET LINE DMC-1 DUPLEX FULL
```

Generally, you use full-duplex mode for local lines and permanently wired telephone lines; you usually use half-duplex mode for dialup remote telephone lines used with half-duplex synchronous modems. If you use a modem, consult the manufacturer's documentation for full- or half-duplex characteristics.

3.6.3.3 Line Timers

Line timers control the frequency of message retransmission at the DDCMP level. There are six line timers:

- Service timer—sets the maximum amount of time allowed to elapse before a retransmission is necessary when service operations are under way.
- Retransmit timer—sets the maximum amount of time allowed to elapse before a retransmission is necessary on a multipoint line. This is the amount of time a control station will wait for a tributary to respond. For a DMF32 tributary, it is the maximum amount of time the tributary will hold the line before returning control to the control station. For an X.25 line, it is the maximum amount of time before a frame is retransmitted (see Section 3.6.5.1).
- Dead timer—sets the amount of time between polls of the dead tributaries.
- Delay timer—sets the amount of time to delay between polls of active tributaries.
- Scheduling timer—sets the time limit between recalculations of tributary polling priorities.
- Stream timer—sets the amount of time that a tributary or half-duplex remote station is allowed to hold the line.

The DMP11 automatically handles message retransmission for normal operations. However, when a DDCMP circuit is in the SERVICE state, a line retransmission timer is necessary because the DMP11 does not handle retransmission in maintenance operation protocol (MOP) mode.

You can determine the optimum value to use for the retransmit timer for a synchronous DDCMP line. The formula involves a constant obtained from the calculation of the time required for transmission and reception of the contents of a single executor buffer over the line. To derive the constant, multiply the executor buffer size (in bytes) by 8 bits/byte, divide the result by the line speed (in bits per second), and multiply by 2 (for transmission and reception). To this result, add a factor representing the time allotted for transmission delay and processing overhead (for DDCMP lines, a factor of 1/2 is used). Convert the final value to milliseconds by multiplying by 1000 ms/sec. When the constants are multiplied out, the remaining constant is 20,000, which applies to the following retransmit timer calculation:

```
RETRANSMIT TIMER = (20000 * buffer-size * number-of-buffers)/bps-of-line
```

In general, use this formula to calculate the best value for the retransmit timer (in milliseconds).

The number of buffers is the value specified for the MAXIMUM TRANSMITS parameter in the SET CIRCUIT command; it represents the maximum number of data messages that the tributary can transmit in a single poll interval (see Section 3.5.3.2).

Assume an example with an executor buffer size of 576, a line of 56K bits per second (bps), and four buffers per selection interval. The formula would be calculated as follows:

```
RETRANSMIT TIMER = (20000 * 576 * 4)/56000 = 820 milliseconds
```

To set a retransmit timer for a DDCMP line, use the RETRANSMIT TIMER parameter, as follows:

```
NCP>SET LINE DMP-2 RETRANSMIT TIMER 820
```

This command sets the retransmission frequency for line DMP–2 to 820 milliseconds. If a message is not acknowledged in 820 milliseconds, it is retransmitted.

The preceding formula does not apply to the DMF32 tributary mode. The value of the retransmit timer is the maximum time the tributary will hold the line before returning control to the control station. For DMF32 tributary mode, therefore, the more active the tributary, the higher the value to which you should set the retransmit timer (a value of 2000 is recommended). For inactive tributaries, set the timer value lower (a value of 500 milliseconds is recommended).

3.6.3.4 Satellite Transmission Control

For a connection over a very long path, such as a satellite link, use the TRANSMIT PIPELINE parameter to establish the maximum number of DDCMP messages that may be transmitted over a DMR11 line without waiting for a positive acknowledgment from the remote node. This parameter is useful for satellite transmissions because of the long round-trip delay between message transmission and acknowledgment. For example, the following command sets a maximum of 10 DDCMP messages for the line DMC-2:

```
NCP>SET LINE DMC-2 ... TRAMSMIT PIPELINE 10
```

The TRANSMIT PIPELINE parameter is optional and, by default, takes the value 7.

Because of the system memory overhead involved, you should avoid arbitrarily setting this value higher than necessary. The optimum value for the TRANSMIT PIPELINE parameter for the DMR11 is equal to the number of DDCMP messages that can be transmitted before the first message in the sequence is acknowledged. You can calculate the optimum TRANSMIT PIPELINE value using the following algorithm:

```
messages = (delay*rate)/(size*8)
```

where:

delay — Is the link's round trip delay time in seconds, which is the total time required for a message to reach the remote receiver and for the acknowledgment to be received by the transmitter. You can determine the delay value from information supplied by the carrier providing the leased circuit, or by observing the delay on suitable line-monitoring equipment.

rate — Is the line speed in bits per second.

size — Is the average DDCMP message size in bytes, which can be calculated by dividing the number of bytes transmitted by the number of messages transmitted. Use the SHOW LINE command with the COUNTERS parameter to determine the number of bytes and messages transmitted. Line counters are described in Section 3.6.6.

For example, to determine the optimum TRANSMIT PIPELINE value of a satellite link that has a round trip delay of 0.67 seconds, a line speed of 9.6K bits per second, and an average DDCMP message size of 40 bytes, calculate the following:

```
(0.67*9600)/(40*8) = 20
```

For this example, the optimum value for TRANSMIT PIPELINE is 20 messages.

3.6.3.5 Asynchronous DDCMP Line Parameters

The LINE SPEED, HANGUP, and SWITCH parameters apply only to asynchronous DDCMP lines. Values for these parameters are provided automatically when a line is switched dynamically from a terminal line to an asynchronous DDCMP line. When you initiate a dynamic connection between two nodes over a telephone line, these parameters are included in the line entries NETACP supplies to the NCP database. For static asynchronous DDCMP lines, these parameters usually assume their default values.

The LINE SPEED parameter specifies in baud the speed of an asynchronous DDCMP line. The parameter defaults to the current speed of the line. If two asynchronous lines are connected, both lines must have the same line speed. If a dynamic connection is made, this value is supplied automatically for each line. For a static asynchronous line, the default line speed value is the value of the /SPEED qualifier in the DCL command SET TERMINAL you specified to cause the terminal line to be converted to an asynchronous line.

The HANGUP parameter determines whether the modem signal is dropped when the line is shut down. When you shut down a dynamically switched asynchronous line, the modem carrier is dropped if the value of the parameter is HANGUP ENABLED. This value, supplied automatically, corresponds to the /HANGUP qualifier in the SET TERMINAL command you specified to cause the terminal line to be switched to an asynchronous line. If the value supplied for the parameter is HANGUP DISABLED, the modem signal is not dropped when the line is shut down. For a static asynchronous line, the parameter defaults to HANGUP ENABLED.

The SWITCH parameter indicates whether an asynchronous DDCMP line is to be switched back to a terminal line after it is disconnected from the network (when the channel to the network is deassigned). The SWITCH parameter is enabled automatically for a dynamic asynchronous line so that the line can be switched back to a terminal line when the dynamic connection is broken. The parameter defaults to SWITCH DISABLED for a static asynchronous line, which remains available as a communications line even when not assigned a channel to the network. Generally, you do not need to set the SWITCH parameter manually.

3.6.4 Ethernet Line Parameters

Ethernet lines support no service functions. Parameters the Ethernet lines have in common with other DECnet lines are the COUNTER TIMER, PROTOCOL, STATE, and BUFFER SIZE parameters. You can use the BUFFER SIZE parameter to optimize performance over a high-speed data link such as an Ethernet (see Section 3.6.2.2).

The Ethernet address associated with the Ethernet line device hardware is displayed as a read-only parameter, HARDWARE ADDRESS, in response to the SHOW LINE command. For example:

```
NCP>SHOW LINE UNA-0 CHARACTERISTICS
```

This command results in the following information being displayed for UNA–0:

```
Protocol          = Ethernet
Hardware address  = AA-00-03-00-00-0C
```

See the discussion of Ethernet physical addresses in Section 2.1.1, and the general description of the format of Ethernet addresses in Section 3.3.4.

3.6.5 X.25 Line Parameters

All X.25 lines must specify the LAPB or LAPBE protocol. The line parameters unique to X.25 lines include the INTERFACE, HOLDBACK TIMER, MAXIMUM RETRANSMITS, MAXIMUM BLOCK, MAXIMUM WINDOW, and NETWORK parameters.

3.6.5.1 Frame Control for X.25 Lines

The MAXIMUM BLOCK and MAXIMUM RETRANSMIT parameters control the size and transmission of frames over an X.25 line; the RETRANSMIT TIMER and MAXIMUM RETRANSMIT parameters control the retransmission of frames. The MAXIMUM WINDOW parameter controls the number of frames for which outstanding acknowledgments are allowed. The HOLDBACK TIMER parameter controls the acknowledgment of frames.

Use the RETRANSMIT TIMER parameter to control the frequency of frame retransmission at X.25 level 2 on LAPB and LAPBE lines. For example, the following command sets the retransmission frequency to 2000 milliseconds for the line DUP–0:

```
NCP>SET LINE DUP-0 ...  RETRANSMIT TIMER 2000 ...
```

In other words, if a frame is not acknowledged in 2000 milliseconds, it is retransmitted.

The value for this parameter depends on the size of the frame and the speed of the X.25 line; refer to the *Public Network Information* manual for recommended values. Specify a value in the range 1 to 65,535. The value must not be smaller than twice the value of the HOLDBACK TIMER parameter, if one is set.

The RETRANSMIT TIMER parameter is optional and, by default, takes the network value. Refer to the *Public Network Information* manual for the network value of this parameter.

To specify the maximum number of times a frame is retransmitted over a specified X.25 line, use the MAXIMUM RETRANSMITS parameter. For example, the following command indicates that if a frame is not acknowledged in 2000 milliseconds it is retransmitted and that this operation is to be performed a maximum of 10 times:

```
NCP>SET LINE DUP-0 ... RETRANSMIT TIMER 2000 -
_ MAXIMUM RETRANSMITS 10 ...
```

Specify a value in the range 1 to 255.

The MAXIMUM RETRANSMITS parameter is optional and, by default, takes the network value. Refer to the *Public Network Information* manual for the network value of this parameter.

To specify the maximum size of the frame for a particular X.25 line, use the MAXIMUM BLOCK parameter. For example, the following command sets the size of the frame to 133 bytes for the line DUP–0:

```
NCP>SET LINE DUP-0 ...  MAXIMUM BLOCK 133 ...
```

Produce a limit for the frame size, as follows:

1. Calculate the maximum packet size and add 5 bytes.
2. If you subscribe to the LAPBE protocol, add 1 byte to this total.
3. If you subscribe to extended sequence numbering at level 3, add another byte to the total.
4. If you subscribe to the fast select facility and the current total is less than 213 bytes, replace the current total with 213 bytes.

The maximum frame size you specify should be greater than the total you calculated and less than or equal to 4103 bytes.

Note that some communications devices limit the frame size. This may cause errors, even if you follow the preceding rules. For more details, refer to the VAX PSI documentation set.

The MAXIMUM BLOCK parameter is optional and, by default, takes the network value. Refer to the *Public Network Information* manual for the network value of this parameter.

To specify the maximum number of frames for which there are outstanding acknowledgments for a particular X.25 line, use the MAXIMUM WINDOW parameter. For example, the following command sets the maximum to 2 for the line DUP–0:

```
NCP>SET LINE DUP-0 ...  MAXIMUM WINDOW 2 ...
```

The MAXIMUM WINDOW parameter is optional and, by default, takes the network value. Refer to the *Public Network Information* manual for the network value of this parameter.

To specify the maximum time to delay acknowledgments, use the HOLDBACK TIMER parameter. For example, the following command sets the maximum delay to 200 milliseconds for line DUP-0:

```
NCP>SET LINE DUP-0 ...  HOLDBACK TIMER 200 ...
```

The HOLDBACK TIMER parameter is optional. If you do not specify it, acknowledgments are not delayed at all.

Specify a value for this parameter in the range 100 to 32767. The value must not be greater than one-half the value of the RETRANSMIT TIMER parameter.

3.6.5.2 Receive Buffers for X.25 Lines

You can optionally specify the number of buffers in the receive queue of any X.25 lines. Use the RECEIVE BUFFERS parameter, for example:

```
NCP>SET LINE DUP-0 ...  RECEIVE BUFFERS 4 ...
```

Specify a value in the range 2 to 32. By default, the number of buffers is 3. This value is normally adequate for DUP, DPV, and DMF lines. For KMX, KMY, and KMV lines, a value of 8 is recommended.

3.6.5.3 Interface of X.25 Lines

The INTERFACE parameter specifies whether the line is to operate as a DCE or as a DTE. It can take the values DTE or DCE. For example:

```
NCP>SET LINE DUP-0 ... INTERFACE DCE
```

The default is DTE.

Note that you can use the DCE interface only in conjunction with the ISO8208 network profile.

3.6.5.4 Network for X.25 Lines

The NETWORK parameter specifies the network to which the line connects. For example:

```
NCP>SET LINE DUP-0 ... NETWORK TELENET1
```

If more than one network is available, this parameter is mandatory.

3.6.6 Line Counters

DECnet software automatically maintains statistics for certain lines in the network. These statistics are known as line counters. Line counters for DDCMP lines include the number of bytes and data blocks sent and received; local and remote process errors; and the amount of time since the counters were last zeroed. DECnet–VAX currently maintains these counters only for DMP11 and DMF32 lines. Line counters for Ethernet lines include the number of bytes, multicast bytes, data blocks, and multicast blocks sent and received; the number of blocks deferred or sent after collision; and the number of send failures and discarded frames. This counter information may be useful alone or in conjunction with logging information to measure the performance and throughput for a given line. Refer to Section 2.9 for a discussion of logging.

VAX PSI automatically maintains statistics for X.25 lines in the network. These statistics are also called line counters, but are used for X.25 lines only. Such information may include bytes and data blocks sent and received; inbound and outbound data errors; and remote and local reply timeouts, buffer errors, and process errors. These counters, together with component **characteristics**, are useful in monitoring the activity of X.25 lines. The counters may, for example, be employed to measure the performance and throughput of a given X.25 line.

You can use NCP to affect the frequency with which counters are logged and when the counters are zeroed. At any point while the network is running, you can also display line counter statistics using the SHOW LINE COUNTERS command.

To set a timer whose expiration automatically causes the line counters to be logged at the logging sink (location) and then zeroed, use the SET LINE command with the COUNTER TIMER parameter. The following command causes a line counter logging event to take place every 600 seconds:

```
NCP>SET LINE DMC-0 COUNTER TIMER 600
```

To clear this parameter, enter the following NCP command:

```
NCP>CLEAR LINE DMC-0 COUNTER TIMER
```

At any point when the network is running, you can zero line counters for a given line or for all known lines. Enter the following commands to zero line counters:

```
NCP>ZERO LINE DMC-0 COUNTERS
NCP>ZERO KNOWN LINES COUNTER
```

3.7 Routing Commands

As network or system manager, you can use certain NCP command parameters to specify how the network is to be configured into routing and nonrouting nodes and into areas. Other NCP parameters indirectly control the path data takes through the network, and control the timing of routing messages; these parameters have reasonable default values for most networks. If the network is very large, having a network manager rather than individual system managers to be responsible for controlling the flow of data through the network may be helpful.

3.7.1 Specifying the Node Type

You specify the type of node in the TYPE parameter of the DEFINE EXECUTOR command. DECnet–VAX supports three values for the node type: NONROUTING IV, ROUTING IV, and AREA. The type of a Phase IV end node is NONROUTING IV, the type of a Phase IV level 1 router is ROUTING IV, and the type of a Phase IV level 2 router is AREA. For example, to designate the executor as a Phase IV nonrouting node (end node), enter the following command:

```
NCP>DEFINE EXECUTOR TYPE NONROUTING IV
```

To specify the executor as a level 2 router in an area network configuration, enter the following command:

```
NCP>DEFINE EXECUTOR TYPE AREA
```

The default value for node-type depends on the particular type of DECnet–VAX license key registered (see Section 2.4.1.3). If the key is for a full function license (supporting routers and end nodes), the default value of the TYPE parameter is ROUTING IV; if the key is for an end node license, the default (and only value possible) is NONROUTING IV.

Note that you cannot change the executor node type while DECnet is running. You must shut down the network, use the DEFINE command to change the executor node type, and then restart the network.

The SHOW EXECUTOR CHARACTERISTICS command displays the node type of the executor node. The SHOW NODE STATUS command displays the node type of a specified adjacent node. The possible values for the node type are AREA, ROUTING IV, NONROUTING IV, ROUTING III, NONROUTING III, or PHASE II. A Phase IV node can be a level 2 router (AREA), a level 1 router (ROUTING IV), or an end node (NONROUTING IV). A Phase III node can be either a router (ROUTING III) or an end node (NONROUTING III).

3.7.2 Specifying the Area Number in a Node Address

To configure a network for area routing, assign each node to a specific area that has a unique number. The area number is a decimal number, in the range 1 through 63, which appears as a prefix on the decimal node number of the individual node. The node number must be unique within the area. The maximum value for node number is 1023. The area number and the node number are separated by a period. The format of a node address in an area network is as follows:

area-number.node-number

For example, node 300 in area 40 has a node address of 40.300.

To set the node address for the local node in an area configuration, use the SET EXECUTOR command with the ADDRESS parameter, as follows:

```
NCP>SET EXECUTOR ADDRESS 40.300
```

Configuration of a network requires that each node is assigned a node address containing an appropriate area number. If you do not specify an area number in a node address, the executor area number is used.

You can convert a Phase IV node address to a decimal equivalent for use in DCL commands, such as COPY, and in sending messages using the Mail Utility. The algorithm to convert the address to its decimal equivalent is as follows:

```
(area-number * 1024) + node-number
```

You can also convert the address to its hexadecimal equivalent for incorporation in the Ethernet physical address of the node (see Section 3.3.4).

Referring to a node by name is generally more convenient.

3.7.3 Setting Routing Configuration Limits

During configuration of the network, you can establish certain limits related to routing over the network. You can limit the number of routers allowed on a single Ethernet and the number of routing and end nodes permitted on all Ethernet circuits to which the local node is attached. If the network is grouped into areas, you can limit the number of areas allowed.

3.7.3.1 Maximum Number of Ethernet Routers and End Nodes Allowed

Certain NCP command parameters limit the number of routers and end nodes that can be configured on Ethernet circuits. Use the SET CIRCUIT command with the MAXIMUM ROUTERS parameter to set the maximum number of routers permitted on a particular Ethernet circuit. The largest number of routers allowed on an Ethernet is 33, which is the default value of the MAXIMUM ROUTERS parameter. Note that the recommended limit on the number of routers on a single Ethernet circuit is 10, because of the control traffic overhead (routing messages and system identification messages) involved. For example, the following command specifies that no more than five routers can exist on Ethernet circuit UNA-0:

```
NCP>SET CIRCUIT UNA-0 MAXIMUM ROUTERS 5
```

Use the SET EXECUTOR command with the MAXIMUM BROADCAST ROUTERS parameter to specify the maximum number of routing nodes that will be permitted on all Ethernet circuits to which the local node is attached. Each routing node can be either a level 1 router (capable of routing within its own area, if area routing is specified) or a level 2 router (capable of routing within its own area and outside of its area). For example, the following command specifies that a maximum of 12 routers is allowed on Ethernet circuits to which the executor node is connected:

```
NCP>SET EXECUTOR MAXIMUM BROADCAST ROUTERS 12
```

The default value of this parameter is 32.

Use the SET EXECUTOR command with the MAXIMUM BROADCAST NONROUTERS parameter to set the maximum number of nonrouting nodes (end nodes) permitted on all Ethernet circuits to which the local node is attached. For example, the following command specifies that no more than 20 end nodes can exist on all Ethernet circuits to which the executor node is connected:

```
NCP>SET EXECUTOR MAXIMUM BROADCAST NONROUTERS 20
```

The default value is 64.

3.7.3.2 Maximum Number of Areas Allowed

When configuring an area network, use the SET EXECUTOR command with the MAXIMUM AREA parameter if you want to set a limit on the number of areas that the executor node's Routing layer will recognize. For example, if you want a maximum of 50 areas to be recognized, enter the following command:

```
NCP>SET EXECUTOR MAXIMUM AREA 50
```

If you do not specify this parameter, the Routing layer recognizes up to 63 areas.

3.7.4 Routing Control Parameters

NCP supports routing parameters that provide for circuit cost control (COST), control of the total path between any two nodes (MAXIMUM COST, MAXIMUM HOPS), route-through control (MAXIMUM VISITS), and equal cost path splitting (MAXIMUM PATH SPLITS and PATH SPLIT POLICY). For a network divided into areas, the area routing parameters for maximum cost and length of the paths between areas in the network (AREA MAXIMUM COST, AREA MAXIMUM HOPS) also apply. These parameters are used to control the path that data is likely to take when being transmitted through the network, and to minimize congestion at particular nodes in the network. For most networks, the default values for these parameters should be acceptable.

3.7.4.1 Circuit Cost Control Parameter

Figure 3–2 illustrates sample circuit costs attributed to the network example. The following paragraphs discuss routing control parameters as they relate to Figure 3–2.

The COST parameter in the SET CIRCUIT command specifies the circuit cost. For example, the following command sets a cost for the circuit connecting node BOSTON to node NYC:

```
NCP>SET CIRCUIT DMC-2 COST 1
```

This command sets a low cost for the circuit. Numbers in the range of 1 to 25 are valid circuit costs. The default value is 10.

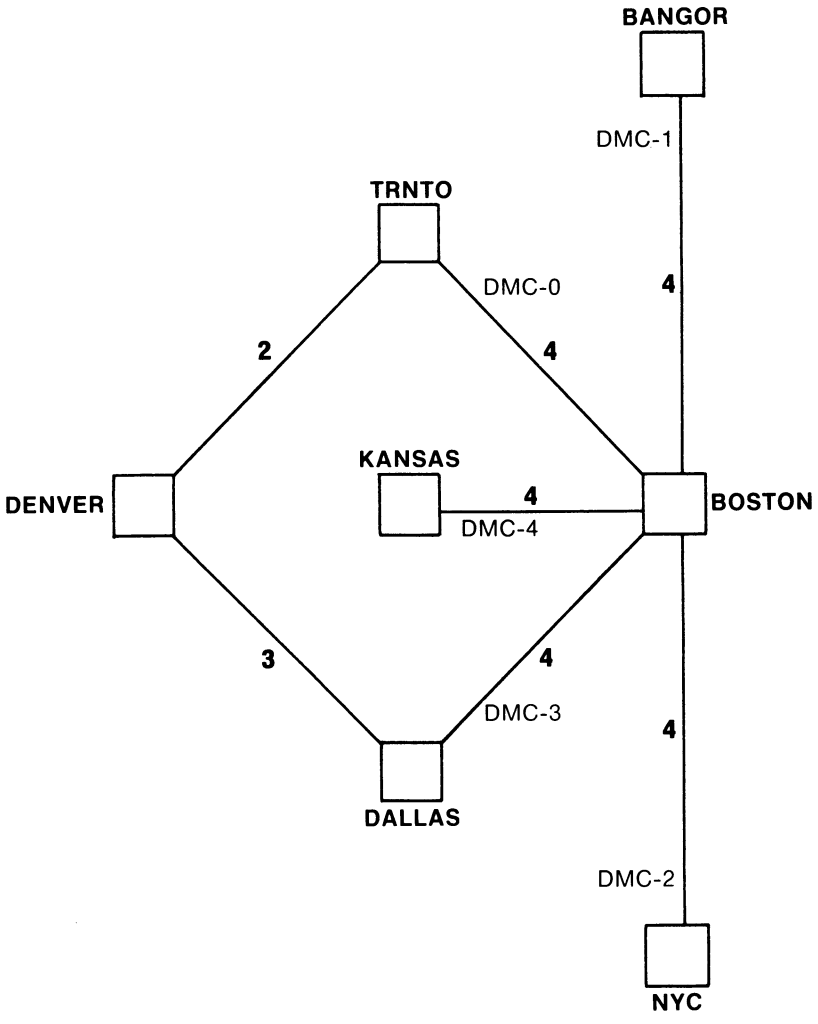
Establishing a circuit cost standard that is uniform across the entire network is recommended. The following algorithm is used to determine appropriate circuit costs. The algorithm is based on circuit delay. Delay is based on circuit bandwidth.

1 where the bandwidth is greater than 100K bits per second.

x where x is approximately equal to 100,000 divided by the bandwidth, and where the bandwidth is greater than 4K bits per second but less than 100K bits per second.

25 where the bandwidth is less than 4K bits per second or the circuit is an X.25 circuit.

Figure 3–2 Network Circuit Costs



3.7.4.2 Maximum Path Control Parameters

You set both the maximum cost for all circuits to the destination node (MAXIMUM COST) and the maximum hops that a packet can make when routed to the destination node (MAXIMUM HOPS) using the SET EXECUTOR command. You use these parameters to ascertain whether a destination is reachable. The value of the MAXIMUM HOPS parameter should always be equal to or greater than the longest possible path within the network. For the network example, a maximum hop parameter value of 6 is sufficient. You should choose the maximum cost and hops values carefully, with regard to the intended use of the network, the actual network configuration, and possible failures. The default values for these parameters are reasonable for most networks.

The following example indicates the use of the SET EXECUTOR command to specify the maximum cost and hops allowed for network routing:

```
NCP>SET EXECUTOR MAXIMUM COST 100 MAXIMUM HOPS 6
```

Values in the range 1 to 1022 are valid for the MAXIMUM COST parameter; the default value is 1022. Values in the range 1 to 30 are valid for the MAXIMUM HOPS parameter; the default value is 30. The value for the MAXIMUM HOPS parameter must be less than or equal to the value for MAXIMUM VISITS. Use as small a number as possible in these ranges.

Figure 3–2 illustrates the relationship between circuit costs and path costs. To send a packet from TRNTO to DALLAS, the system can route it over one of two paths, both of which require two hops; the first path is through BOSTON, the second through DENVER. However, because the path through BOSTON has a cost of 8 and the path through DENVER has a cost of 5, the system routes the packet through DENVER.

Under normal conditions, a MAXIMUM HOPS value of 3 would be sufficient for the network in Figure 3–2. However, if the MAXIMUM HOPS value were set to 3, a failure of the TRNTO-BOSTON circuit would render TRNTO unreachable from NYC, KANSAS, or BANGOR, even though a physical path still exists (the four-hop path NYC-BOSTON-DALLAS-DENVER-TRNTO). Consideration of possible failures is also important in establishing the MAXIMUM COST parameter.

3.7.4.3 Route-Through Control Parameter

The MAXIMUM VISITS parameter in the SET EXECUTOR command specifies the maximum number of nodes a packet can be routed through before arriving at the destination node. For example, the following command sets the number of visits to 12:

```
NCP>SET EXECUTOR MAXIMUM VISITS 12
```

If the number of nodes that the data packet visits exceeds the value of MAXIMUM VISITS, the packet is discarded. Generally, use a value that is two or three times the value for the MAXIMUM HOPS parameter. At a minimum, the value for the MAXIMUM VISITS parameter must be equal to or greater than the value for the MAXIMUM HOPS parameter. The maximum value is 63, which is also the default value.

3.7.4.4 Equal Cost Path Parameters

Circuit costs are used by DECnet to determine the optimum path over which data is to be transmitted. DECnet selects the path with the lowest cost. If there are multiple paths of equal cost, and that cost is the lowest cost, the routing of individual data packets, by default, is split among these equal cost paths. This method of equal cost path splitting improves network efficiency by ensuring that multiple equal cost paths are not idle when there is traffic to be routed. The MAXIMUM PATH SPLITS parameter of the SET EXECUTOR command specifies the maximum number of equal cost paths to be used for routing. For example, the following command sets the maximum number of equal cost paths to 2:

```
NCP>SET EXECUTOR MAXIMUM PATH SPLITS 2
```

The default value for MAXIMUM PATH SPLITS is 1.

Equal cost path splitting operates most efficiently for nodes running VMS Version 5.0 communicating with nodes running DECnet–VAX Version 4.6 or higher because these versions of the operating system support out-of-order packet caching. If some nodes in the network are running DECnet–VAX Version 4.5 or lower, out-of-order packet caching is not supported, and any packets received out of order are discarded. Therefore, splitting traffic over all equal cost paths may result in poor network performance.

To control the equal cost path splitting for routing, you can set the executor parameter PATH SPLIT POLICY. By default, PATH SPLIT POLICY is set to NORMAL, which indicates that all traffic is to be split equally over all equal cost paths to a destination node. To restrict the paths used for routing, you can set PATH SPLIT POLICY to INTERIM. The INTERIM value specifies that all traffic is to be split over all equal cost paths while forcing packets for individual network sessions over the same paths to guarantee that packets are received by the destination node in the correct order. For example, the following command specifies that all traffic for all network sessions is to choose the same paths, rather than being split for routing on all equal cost paths:

```
NCP>SET EXECUTOR PATH SPLIT POLICY INTERIM
```

3.7.4.5 Area Path Control Parameters

When a network is divided into areas, the MAXIMUM COST and MAXIMUM HOPS parameters described previously are used to control the path between each pair of nodes within each area. A second set of routing parameters (AREA MAXIMUM COST, AREA MAXIMUM HOPS) is used to control the total cost and length of paths between level 2 routers within the whole network. In effect, these parameters control the total possible path between areas in the network.

The AREA MAXIMUM COST parameter in the SET EXECUTOR command specifies the limit on the total path cost between the local level 2 router and any level 2 router in the network. This value is the maximum cost of circuits on the longest path between level 2 routers. The AREA MAXIMUM HOPS parameter in the SET EXECUTOR command specifies the maximum number of hops that a packet can make between the local level 2 router and any other level 2 router in the network. You use the AREA MAXIMUM COST and AREA MAXIMUM HOPS parameters to determine whether an area is reachable. The default values for these parameters are reasonable. You should select other values carefully, with regard for the level 2 (area) topology of the network.

The following example illustrates the use of the SET EXECUTOR command to specify the maximum cost and hops permitted for routing between level 2 routers in the network:

```
NCP>SET EXECUTOR AREA MAXIMUM COST 500 AREA MAXIMUM HOPS 10
```

Values in the range 1 to 1022 are valid for the AREA MAXIMUM COST parameter; the default value is 1022. Values in the range 1 to 30 are valid for the AREA MAXIMUM HOPS parameter; the default value is 30.

3.7.5 Routing Message Timers

Routing messages exchanged between adjacent nodes contain information about the cost and hops to each node in the network. Routing update messages are sent automatically whenever there is a change in the information (for example, when a line goes down). Nodes that detect the change (for example, nodes at each end of a line that failed) are the first to send routing update messages. The changed routing information then propagates as far as necessary to update all routers.

Routing updates are also sent periodically under control of the routing timers. These periodic transmissions ensure that routing tables are kept up to date even in the unlikely event that a routing update message is lost.

You set the timer for transmission of routing messages by using the SET EXECUTOR command. For nodes on non-Ethernet circuits, the timer is called the routing timer. Changing the setting of the routing timer causes additional routing messages to be transmitted to all adjacent nodes from the local node, at a specified interval. For example, the following command sets the frequency of transmission of routing messages to 240 seconds:

```
NCP>SET EXECUTOR ROUTING TIMER 240
```

When this timer expires, the local node sends a routing message to all adjacent nodes. Numbers in the range of 1 to 65,535 are valid for the ROUTING TIMER parameter; the default value is 600. Allowing NETACP to supply the default is recommended.

For a node on an Ethernet circuit, the timer is called the broadcast routing timer. When the timer expires, the local node sends a multicast routing configuration message to all nodes on the Ethernet. For example, the following command sets the frequency of routing message transmissions to 30 seconds:

```
NCP>SET EXECUTOR BROADCAST ROUTING TIMER 30
```

The broadcast routing timer for a node on an Ethernet circuit is set to a much lower value (approximately 30 to 40 seconds) than the routing timer for a node on a non-Ethernet circuit (every few minutes). Ethernet routing messages are sent more often so that full routing messages can be exchanged in case of datagram loss. The default value for this parameter is 40.

3.7.6 CI End Node Circuit Failover

If you configure a VAXcluster that uses the CI as its DECnet datalink to include end nodes as well as routers, you can define a backup circuit in each end node that takes over should the primary circuit connecting the end node to its router fail.

An example is a three-node cluster comprised of one router (R) and two end nodes (E1 and E2). Each end node should have a circuit defined to the router. You can define a second circuit in each end node that connects to the other end node. The backup circuit is defined with a higher cost than the primary circuit, and its state is set to ON. Under normal circumstances, with all three nodes operational, the lower cost circuit (to the router) is used. If the router shuts down, this circuit also shuts down. The backup circuit will become the lowest cost circuit in the ON state, and will be used. The backup circuit allows the end nodes to communicate while the router is absent from the cluster.

If nodes E1, E2, and R have CI port addresses 1, 2, and 3, respectively, you could define this topology in node E1 as follows:

```
NCP>DEFINE CIRCUIT CI-0.3 TRIBUTARY 3 COST 1 STATE ON
NCP>DEFINE CIRCUIT CI-0.2 TRIBUTARY 2 COST 10 STATE ON
```

The first circuit is the primary circuit; the second circuit is the backup circuit.

This technique can be extended to a larger cluster with two routers and several end nodes; in each end node, two circuits of different cost are defined, one to each router. The network could then survive the failure of one router, but not both.

3.8 Logical Link Commands

Use the SET EXECUTOR command to set logical link parameters that define the maximum number of active links permitted and to set the timers that control NSP operation. Use the DISCONNECT LINK command to disconnect links while the network is running.

3.8.1 Maximum Number of Links

When defining parameters for the local node, you may specify the maximum number of logical links that can be active for that node. DECnet–VAX uses this value to determine the size of internal data structures. The following command sets the maximum number of links at 30:

```
NCP>SET EXECUTOR MAXIMUM LINKS 30
```

Note that this value includes both inbound and outbound logical links. In this example, you can have only 15 links if both ends of all links are terminated locally.

If an alias node identifier has been established, you may also specify the maximum number of logical links that can be active at the local node using the alias node identifier. For example, the following command sets the alias maximum links at 40:

```
NCP>SET EXECUTOR ALIAS MAXIMUM LINKS 40
```

When a VAXcluster uses an alias node identifier, two kinds of link (alias node and local node) are possible. These links are controlled by the appropriate parameter, MAXIMUM LINKS or ALIAS MAXIMUM LINKS. When you specify both of these parameters, the upper limit on the number of logical links that use the individual node identifier is reduced. Refer to the *VMS Network Control Program Manual* for information about logical link restrictions.

3.8.2 Disconnecting Logical Links

You can selectively disconnect logical links active on the local node while the network is running. The first of the following commands disconnects link 1827; the second disconnects all links active with all remote nodes:

```
NCP>DISCONNECT LINK 1827

NCP>DISCONNECT KNOWN LINKS
```

Use the SHOW KNOWN LINKS command to obtain link status information, including link addresses, and to verify that links have been disconnected upon entering these commands (see Section 3.3). DECnet–VAX maintains and uses link addresses.

Optionally, you can disconnect a single link or all known links to a particular node. For example, the following NCP command disconnects all links to node TRNTO:

```
NCP>DISCONNECT KNOWN LINKS WITH NODE TRNTO
```

3.8.3 Logical Link Protocol Parameters

A variety of parameters exist for controlling NSP-related logical link activity. These parameters regulate the bounds for NSP connect sequences, inactivity intervals, and message retransmission. Another parameter limits the amount of nonpaged pool NSP uses for logical link transmission. You can change these parameters at any time, without affecting existing logical links.

3.8.3.1 Incoming and Outgoing Timers

There are two timers that regulate NSP connect sequences: an incoming timer and an outgoing timer. Use the INCOMING TIMER parameter to specify the maximum duration between the moment a logical link connection is received for a process on the local node and the moment the process accepts or rejects the connection. Using a value between 30 and 60 is recommended. To allow 30 seconds for connection confirmation, enter the following command:

```
NCP>SET EXECUTOR INCOMING TIMER 30
```

Expiration of this timer signals that a timeout has occurred. In effect, this timer protects the local node against a process that never responds to an inbound connection request.

The OUTGOING TIMER parameter specifies a timeout value for the duration between the time a connection is requested and the time it is acknowledged by the destination node. Using a value between 30 and 60 is recommended. For example, the following command allows 30 seconds to elapse before a timeout is assumed to have occurred:

```
NCP>SET EXECUTOR OUTGOING TIMER 30
```

A typical value for this timer ranges from 10 to 90 seconds, depending on line speed and network diameter. The network diameter is the maximum diameter over the set of shortest paths between all pairs of nodes in the network. In effect, this timer protects the user on the local node against a connection request that never completes.

3.8.3.2 Inactivity Timer

A logical link is inactive when no data is transmitted in either direction for a given interval of time. The inactivity timer regulates the frequency with which local DECnet software tests the viability of an inactive link, thereby protecting the user against a link that may be permanently unusable. Use the INACTIVITY TIMER parameter to specify the maximum duration of inactivity before the local node tests the viability of the link. For example, the following command sets the inactivity interval to 60 seconds:

```
NCP>SET EXECUTOR INACTIVITY TIMER 60
```

When this timer expires, DECnet–VAX generates artificial traffic to test the link. The timer starts after an incoming message for the link has been processed. The timer is reset if any messages are received on the link.

3.8.3.3 NSP Message Retransmission

A third group of parameters regulates the frequency of NSP message retransmission. These are the DELAY WEIGHT, DELAY FACTOR, and RETRANSMIT FACTOR parameters for the local node. Using default values for these parameters is recommended.

NSP estimates the current delay in the round-trip transmission to a node with which it is communicating. The value of the DELAY WEIGHT parameter is used to calculate a new value of the estimated round trip delay. The old round trip delay is weighted by a function of this statistical factor to calculate the new round trip delay. If the delay weight is set high, the retransmit time changes slowly. If the weight is set low, the observed round trip time can change quickly if the observed round trip delays vary widely, and thus the retransmit time can change more rapidly.

The value of the DELAY FACTOR parameter is multiplied by one-sixteenth of the estimated round trip delay time to determine the appropriate value for the time to retransmit certain NSP messages.

You use values in the range of 1 to 255 to specify values for the DELAY FACTOR parameter, as in the following example:

```
NCP>SET EXECUTOR DELAY WEIGHT 3 DELAY FACTOR 48
```

The default value is 80. For a complete discussion of these concepts, refer to the *Network Services Protocol Functional Specification.*

The value of the RETRANSMIT FACTOR parameter regulates the number of times NSP reattempts a transmission when its retransmission timer expires for a logical link. This value must be a number in the range of 1 to 65,535; the default value is 10. For example, the following command specifies that NSP should reattempt a transmission no more than 10 times:

```
NCP>SET EXECUTOR RETRANSMIT FACTOR 10
```

If NSP tries to retransmit an eleventh time, the logical link disconnects.

In the process of logical link connect sequences, the value of the RETRANSMIT FACTOR parameter takes precedence over the OUTGOING TIMER value. As a result, the actual time necessary for the specified number of retransmits may not match the setting of the OUTGOING TIMER parameter.

Note: Unless you have a special need to change the operating characteristics of a logical link, you should use the default values for DELAY WEIGHT, DELAY FACTOR, and RETRANSMIT FACTOR. In other words, do not define these parameters in the permanent database.

3.8.3.4 Pipeline Quota

The PIPELINE QUOTA parameter in the SET EXECUTOR command specifies the maximum number of bytes NCP can use from nonpaged pool to buffer logical-link transmit requests. In effect, this quota determines the number of packets NCP transmits on a single logical link before waiting for a positive acknowledgment from the remote end of the link. You determine the number of packets by dividing the PIPELINE QUOTA value by the EXECUTOR BUFFER SIZE value.

Unlike previous releases of DECnet–VAX, this PIPELINE QUOTA is not deducted from the byte count quota of the user process. This change allows the system manager to set the process byte count quota to sensible values without concern for the nonpaged pool requirements of DECnet. DECnet's nonpaged pool usage with respect to the transmission over logical links is bounded by the product of the values of the PIPELINE QUOTA and MAXIMUM LINKS parameters.

The following command sets a pipeline quota of 6000 bytes for the local node that is using a satellite link:

```
NCP>SET EXECUTOR PIPELINE QUOTA 6000
```

The default value for PIPELINE QUOTA is currently 3000. If satellite communication is being used, you may need to increase this value to 6000 or more in order to improve DECnet performance.

Note: The SET EXECUTOR parameter PIPELINE QUOTA should not be confused with the SET LINE parameter TRANSMIT PIPELINE. The PIPELINE QUOTA parameter relates to transmission over logical links, while TRANSMIT PIPELINE relates to data links. These parameters address different levels of the Digital Network Architecture.

3.9 Object Commands

Use the SET OBJECT command to establish and modify the object parameters listed in Table 3–6. To remove any or all object parameters from the volatile database, use the CLEAR OBJECT command.

Table 3–6 Object Parameters and Their Functions

Parameter Function	Parameter
Identifies object by number	NUMBER number
Identifies command procedure for starting the object	FILE file-id
Specifies connect privileges for user-level access control	PRIVILEGES privilege-list

Table 3–6 (Cont.) Object Parameters and Their Functions

Parameter Function	Parameter
Specifies optional default proxy login access control for the object	PROXY { INCOMING | OUTGOING | BOTH | NONE }
Determines how the object will respond to incoming connect requests directed to the alias node identifier	ALIAS INCOMING { DISABLED | ENABLED }
Associates outgoing connect requests for the object with the alias node identifier	ALIAS OUTGOING { DISABLED | ENABLED }
Specifies optional default access control for inbound connects	ACCOUNT account PASSWORD password USER user-id

3.9.1 DECnet–VAX Objects

Use the SET OBJECT command to establish and modify certain DECnet–VAX objects and their command procedures.

3.9.1.1 DECnet–VAX Object Identification

When defining or modifying object parameters, you must identify the name of the object. DECnet object names are descriptive alphanumeric strings of up to twelve characters. DECnet software also uses object numbers as unique object identifiers. Object numbers have a range of 1 to 255. Most user-defined images have a 0 object type. However, a user program should have a nonzero number assigned when it provides a known service. You may define an object name of TASK in the configuration database to objects with a 0 object type if you provide additional required privileges or default inbound access control to the object.

Generic objects such as FAL and NML have nonzero object numbers that are recognized throughout the network. User-defined images may have unique nonzero object numbers; numbers between 128 and 255 are reserved for this purpose. (For a list of object numbers and their associated names, refer to the *VMS Network Control Program Manual*.) Unlike objects with a 0 object type, you must set each nonzero object in the configuration database. Use the NUMBER parameter to specify a unique object number for nonzero objects. For example:

```
NCP>SET OBJECT FOO NUMBER 129
```

Note that the object name may not be unique to the generic services specified. Only object numbers are unique across systems. For consistency, however, using object names as they are normally referenced throughout the network is recommended.

When NETACP receives a logical-link connect request message from a remote node, it translates the message into network connect block (NCB) format and delivers it to the destination object running on the local node. (Refer to Chapter 8 for a description of the NCB.)

3.9.1.2 Using the Cluster Alias Node Identifier for the Object

Command parameters for the SET OBJECT and DEFINE OBJECT commands specify how certain objects treat incoming and outgoing connect requests associated with the alias node.

By specifying the ALIAS OUTGOING parameter for a particular object, you can indicate whether the object uses the alias node address in any outgoing connect request.

This parameter makes it possible to direct an object such as MAIL to use the alias node address rather than the executor address for outgoing connections. For example, to direct the object FOX to use the alias node identifier for all outgoing connect requests, enter the following command:

```
NCP>SET OBJECT FOX ALIAS OUTGOING ENABLED
```

By default, only the object MAIL is so enabled. All other objects are disabled unless specified as otherwise.

Objects such as PHONE, which use a protocol that depends on multiple links, should not have the ALIAS OUTGOING parameter enabled.

Use the ALIAS INCOMING parameter to specify how certain objects are to respond to incoming connect requests that are directed to the alias node. You can either enable or disable specific objects from receiving these incoming connections.

This parameter allows you to restrict incoming connections to only those objects that are appropriate. You should not enable any object that can receive multiple incoming links or whose resources are not available clusterwide. For example, to disallow the object FOO from receiving incoming connect requests directed to the alias node address, enter the following command:

```
NCP>SET OBJECT FOO ALIAS INCOMING DISABLED
```

By default, if you establish an alias node identifier for the node, ALIAS INCOMING is enabled for all objects except PHONE. If a user attempts to use an alias node address to connect to an object for which ALIAS INCOMING has been disabled, the status message NO SUCH OBJECT is returned.

3.9.1.3 Example of Using the Cluster Alias Node Identifier

The following scenario illustrates how use of an alias node identifier can facilitate communication between a node within a cluster and a remote node.

A cluster includes nodes THRUSH and ROBIN. The network manager establishes a node name CLUSTR in the database by entering the following DEFINE NODE command:

```
NCP>DEFINE NODE 2.13 NAME CLUSTR
```

To establish the node name CLUSTR as the alias node identification for the cluster, the network manager then enters the following command:

```
NCP>DEFINE EXECUTOR ALIAS NODE CLUSTR
```

Because an alias node identifier has been set, the ALIAS INCOMING parameter is enabled by default. This means that all incoming connect requests addressed to the alias node identifier are routed to a node that uses the alias.

The network manager also indicates that the MAIL object is to use the alias node identifier in its outgoing connect requests by entering the following command:

```
NCP>DEFINE OBJECT MAIL ALIAS OUTGOING ENABLED
```

After the network is started, a user with the user name JONES logs on to node THRUSH. JONES then sends a mail message to user SMITH on node BOSTON, which is outside the cluster. Because MAIL is enabled for outgoing connect requests, it appears that JONES has sent mail from node CLUSTR. An hour later, when user SMITH reads the mail from JONES, the mail is associated with the node-identifier CLUSTR::JONES.

SMITH decides to reply to the mail from JONES. SMITH sends the mail message to JONES using the destination node CLUSTR.

Meanwhile, the node THRUSH has been taken down for maintenance, so JONES has logged on to node ROBIN. Because ROBIN has also been enabled for incoming connect requests addressed to the alias node identifier, JONES receives the mail from SMITH. The mail is addressed to CLUSTR::JONES, and is delivered to a node that uses the alias.

3.9.1.4 DECnet–VAX Command Procedure Identification

For nonzero-numbered objects, the default name of this command file is SYS$SYSTEM:*objectname*.COM. Nonzero objects are identified in the logical link connect message only by object number. Therefore, there must be an entry in the object volatile database that enables NETACP to locate the object name using the object number as a key. When you install DECnet–VAX, nonzero object network-defined command procedures are entered by default in the SYS$SYSTEM directory, and NETACP knows about these command procedures. The supplied command files, named *objectname*.COM, include FAL, HLD, NML, EVL, DTR, MAIL, PHONE, and MIRROR. Except for those command procedures supplied by DIGITAL, you must create a command procedure for every object that can be started by an inbound connection request. You should name command procedures for nonzero objects *objectname*.COM and place them in SYS$SYSTEM.

For zero-numbered objects, the default name of this command file is SYS$LOGIN:*objectname*.COM. Zero objects are identified in the logical link connect message by object name. Therefore, there is no need for an entry in the object volatile database. You can, of course, specify an entry in the object database at any time. You are required to include a separate entry if you want special features such as default inbound access control information.

In either case, you can override the rules for locating the command file by explicitly specifying a command procedure file in the SET OBJECT command. This file is associated with the object in the object volatile database, as shown in the following example:

```
NCP>SET OBJECT FAL NUMBER 17 FILE SYS$MANAGER:TRIALFAL.COM

NCP>SET OBJECT USERS NUMBER 0 FILE SYS$SYSTEM:USERS.COM
```

This technique can be particularly useful for zero-numbered objects. The command file would then be found in the same place, regardless of which access control information you use. If you do not specify the FILE parameter, copies of the command file would have to exist in the SYS$LOGIN directory of every account in which the object may possibly run.

Note: Because REMACP is started by a RUN command in RTTLOAD.COM, there is no REMACP.COM procedure to start the object, and the software does not create a REMACP.LOG file.

You can also invoke an image directly to serve as a network object, rather than using a command procedure. To do this, specify the object file name as *objectname*.EXE, as in the following example:

```
NCP>SET OBJECT FAL NUMBER 17 FILE FAL.EXE
```

You should place the image in SYS$SYSTEM. This approach causes the object to be started up more quickly; it is useful in cases where no advantage is gained by invoking the image from a command procedure. The session log appears as part of the NETSERVER.LOG file.

3.9.2 VAX PSI Objects

Use the SET OBJECT command to identify each VAX PSI object, its command procedure, and the account information to be used by calls coming in to the object from remote DTEs.

3.9.2.1 VAX PSI Object Identification

Each object must have a unique name. Object names are descriptive alphanumeric strings up to 12 characters in length, for example, OBJONE.

For VAX PSI objects, you must specify NUMBER 0 the first time you specify the object.

Use the SET OBJECT command to identify the object. For example:

```
NCP>SET OBJECT OBJONE NUMBER 0 ...
```

3.9.2.2 VAX PSI Command Procedure Identification

The system manager must create a command procedure for every object that can be accessed by a destination. Command procedures are named *filename.COM*.

Use the FILE parameter to specify the command procedure for an object. For example:

```
NCP>SET OBJECT OBJONE ...  FILE STARTUP.COM ...
```

The file name is associated with the object identification in the configuration database. To allow connections to an object, you must first create a command procedure in the default directory for the user account. VAX PSI automatically creates a log file (*filename*.LOG) every time an incoming call causes an object's command file to be executed. This file is created in the default directory of the account. The log file is helpful for debugging your own network tasks when an error occurs.

The command procedure contains at least a RUN command for an image. It may also contain terminal assignments for debugging purposes (for example, DBG$INPUT and DBG$OUTPUT). There are no restrictions on the type of commands you can have in this file.

3.9.2.3 VAX PSI Object Account Information

You should specify the account information that is used by incoming calls from remote DTEs for each object.

Use the ACCOUNT, PASSWORD, and USER parameters to specify the account information. For example:

```
NCP>SET OBJECT OBJONE ... USER NET -
_ PASSWORD NET ACCOUNT PAULS...
```

The USER parameter is mandatory. The ACCOUNT and PASSWORD parameters are optional and, by default, are not used.

3.10 X.25/X.29 Server Module Commands

The X25-SERVER and X29-SERVER module components handle incoming X.25 and X.29 calls from a PSDN. The server components contain records that identify destinations for incoming calls and associate, with each destination, parameters that determine whether the destination can handle an incoming call. The destination can be on a local DTE connected directly to a PSDN, or on a host node to which an X.25 multihost connector node is forwarding incoming calls. The server database also specifies the maximum number of incoming and outgoing circuits that each module (that is, all destinations for that particular module) can have, and specifies the state of the module.

3.10.1 X25–SERVER and X29–SERVER Module Identification

Use the SET MODULE X25-SERVER and SET MODULE X29-SERVER commands to identify the modules that handle incoming calls. The parameters for these two modules are the same. Use separate commands to specify the destination qualifier (DESTINATION), and the module parameters (MAXIMUM CIRCUITS, STATE, and COUNTER TIMER).

3.10.2 Destination Identification

Each destination must have a unique name. Destination names are descriptive alphanumeric strings, from 1 to 16 characters in length. Use the DESTINATION qualifier to specify the destination name. For example:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE ...
```

Associate any of the following parameters with each destination: SUBADDRESSES, GROUP, CALL MASK, CALL VALUE, CALLED ADDRESS, INCOMING ADDRESS, RECEIVING DTE, REDIRECT REASON, SENDING ADDRESS, NETWORK, EXTENSION MASK, EXTENSION VALUE, PRIORITY, OBJECT, and NODE.

You use the parameters from SUBADDRESSES through EXTENSION VALUE to determine whether a call is handled. (If you do not specify a parameter, the parameter is not used.) The PRIORITY parameter sets the priority of the destination, and the OBJECT parameter names the object activated when a destination accepts an incoming call. The NODE parameter identifies the host node on which the destination is located, if the call is received through a local X.25 multihost connector node.

The parameters are described in the following sections.

3.10.2.1 DTE Subaddress Range

Use the SUBADDRESSES parameter to specify a local DTE subaddress or a range of subaddresses for each destination. The destination uses this information to decide if it can handle incoming calls.

In the following command, destination JOE will handle only incoming X.25 calls that specify local DTE subaddress 35:

```
NCP>SET MODULE X25-SERVER DESTINATION -
_ JOE SUBADDRESSES 35...
```

The following command, however, specifies that destination JOE will handle all incoming X.25 calls that specify a local DTE subaddress in the range 12 to 24:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE -
_ SUBADDRESSES 12-24...
```

A subaddress is a decimal integer in the range 0 to 9999. Separate two subaddresses with a single hyphen to indicate a range. The second subaddress must always be greater than the first.

The SUBADDRESSES parameter is optional, and, by default, no subaddress range is used to determine if the destination can handle an incoming call.

3.10.2.2 Group Identification

Use the GROUP parameter to specify a closed user group (CUG) or bilateral closed user group (BCUG) name for each destination. The destination uses this information to decide if it can handle incoming calls. The following command indicates that destination JOE will handle only incoming X.25 calls that originate from a DTE that is a member of the closed user group ESECUG:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE -
_ GROUP ESECUG ...
```

The GROUP parameter is optional, and, by default, no group name is used to determine if the destination can handle an incoming call.

3.10.2.3 Remote DTE Identification

Use the SENDING ADDRESS parameter to specify the remote DTE address that sent the call, that is, the address contained in the calling address field in the call packet. The DTE address consists of 1 to 15 digits. The destination uses this information to decide if it can handle incoming calls.

As an example, the following command specifies that the destination JOE will handle only incoming X.25 calls that come from the remote DTE with address 987321654:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE -
_ SENDING ADDRESS 987321654...
```

The SENDING ADDRESS parameter is optional and, by default, no remote DTE address is used to determine if the destination can handle an incoming call.

3.10.2.4 User Data Field

Optionally use the CALL MASK and CALL VALUE parameters to specify a call mask (to be applied to incoming call data before it is tested) and a call value (the string used to test incoming call data). If you specify these parameters, VAX PSI extracts the user data from the incoming call request and performs a logical AND operation between this data and the call mask. VAX PSI then compares the result of this operation with the call value; if the fields match, the destination can accept the incoming call. Note that the call mask and the call value you specify must be the same length. For example, the following command indicates that destination JOE will handle only incoming X.25 calls that contain value 11 in their user data fields:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE CALL VALUE 11 -
_ CALL MASK FF ...
```

The CCITT (Comite Consultatif International Telegraphique et Telephonique) recommends that you use a value of 01 for incoming X.29 calls. As an example, the following command indicates that destination JIM will handle only incoming X.29 calls that contain 01 in their user data fields:

```
NCP>SET MODULE X29-SERVER DESTINATION JIM CALL VALUE 01 -
_ CALL MASK FF ...
```

Specify strings of 2 to 32 hexadecimal digits for the two parameters.

You can use these two parameters to further identify X.29 calls when the terminal user first connects to the PAD (Packet Assembly/Disassembly Facility). To do so, specify a destination for the X29–SERVER that recognizes a value entered as call data when connecting to the PAD. For example, the following command indicates that the destination JANE will handle incoming X.29 calls that specify a call data value of A (41 is the hexadecimal value for A):

```
NCP>SET MODULE X29-SERVER JANE CALL VALUE 0000000041 -
_ CALL MASK 00000000FF ...
```

The CALL MASK and CALL VALUE parameters are optional and, by default, no mask or value is used to determine if the destination can handle an incoming call.

3.10.2.5 Address Extension

The address extension facility is intended to support end-to-end signaling, as described in the *VAX P.S.I. Management Guide*. Two parameters are used for address extension: EXTENSION MASK and EXTENSION VALUE. These are used like the CALL MASK and CALL VALUE parameters. The extension mask is applied (by VAX PSI) to the called address extension in an incoming call and the extension value is compared with the result. For example:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE -
_ EXTENSION VALUE 12340000AA EXTENSION MASK FFFF0000FF
```

Here, the destination JOE will handle only incoming calls that have a hexadecimal address extension value of 1234, followed by any four digits, followed by AA.

3.10.2.6 Call Redirection

The call redirection facility allows you to receive incoming calls that have been redirected to your DTE from another DTE connected to the same PSDN. Four parameters are used to match redirected calls: REDIRECT REASON, CALLED ADDRESS, INCOMING ADDRESS and SENDING ADDRESS.

The REDIRECT REASON parameter takes one of the following values:

BUSY	Indicates that the called DTE is busy
OUT OF ORDER	Indicates that the called DTE is of order
SYSTEMATIC	Indicates that calls to the called DTE are automatically rerouted

The CALLED ADDRESS parameter should be matched against the originally called address. This will be found in the call redirection notification facility field of the call packet.

The INCOMING ADDRESS parameter should be matched against the address in the called address field of the call packet.

The SENDING ADDRESS parameter (described in Section 3.10.2.3) should be matched against the address in the calling address field of the call packet.

For example:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE -
_ CALLED DTE 123999456 -
_ INCOMING ADDRESS 123789456 -
_ SENDING ADDRESS 123888456 -
_ REDIRECT REASON BUSY
```

Here, the destination JOE will handle calls sent from 123888456 to 123999456 that have been redirected to 123789456, because 123999456 is busy.

3.10.2.7 Receiving DTE

Use the RECEIVING DTE parameter to specify the local DTE that has received the call. For example:

```
NCP>SET MODULE X25-SERVER DESTINATION READING6 -
_ RECEIVING DTE 234295432 ...
```

This command creates the destination READING6, which matches any calls received on DTE 234295432.

3.10.2.8 Priority

Use the PRIORITY parameter to specify the priority of each destination. If more than one destination can accept an incoming call, the destination with the highest priority is used. For example, the following command assigns a priority of 3 to destination JOE:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE PRIORITY 3...
```

Specify a priority value in the range 0 to 255.

The PRIORITY parameter is mandatory if you specify more than one destination that could handle the same incoming call. Otherwise, this parameter defaults to 0.

3.10.2.9 Object Identification

Use the OBJECT parameter to specify the name of the object that is activated when an incoming call arrives and is accepted by the destination. Specify the name as an id-string. If the object name is a string of digits, enclose the string in quotation marks. You must specify the object itself, using the SET OBJECT command (see Section 3.9.1).

The following command specifies the object OBJONE:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE OBJECT OBJONE...
```

When you specify an X.25 Server destination for the first time, the OBJECT parameter is mandatory unless a NODE parameter is specified. If the destination includes a NODE parameter, the OBJECT parameter defaults to 36.

3.10.2.10 Host Node Identification

Use the NODE parameter with the DESTINATION qualifier to identify a host node on which the destination is located. You can specify the NODE parameter only for X.25 Server destinations. A host node receives X.25 calls that have been forwarded by an X.25 connector node connected directly to a PSDN. If your local DECnet–VAX node is configured with VAX PSI software in multihost mode to serve as an X.25 connector node, you must specify the NODE parameter for each host destination. For example, if your local node is a connector node, enter the following command to specify that incoming calls are to be forwarded to the indicated destination on host node THRUSH on the same Ethernet:

```
NCP>SET MODULE X25-SERVER DESTINATION THRUSH -
_ SUBADDRESSES 1-10 OBJECT 36 NODE THRUSH
```

Note that you must configure the host node THRUSH with VAX PSI Access software to be associated with the X.25 connector node. You must enter the following command at node THRUSH to specify the destination of X.25 calls being forwarded by the X.25 connector node:

```
NCP>SET MODULE X25-SERVER DESTINATION JOE -
_ SUBADDRESSES 1-10 OBJECT OBJONE PRIORITY 1
```

3.10.3 Maximum Circuits

Use the MAXIMUM CIRCUITS parameter to specify the maximum number of circuits that the module (that is, all destinations) can handle. The following command enables the X.25 Server module (the call handler for incoming X.25 calls) to handle a maximum of 32 circuits (incoming and outgoing calls) at any one time:

```
NCP>SET MODULE X25-SERVER MAXIMUM CIRCUITS 32...
```

The MAXIMUM CIRCUITS parameter is optional and, by default, the maximum is 255.

3.10.4 Operational State of Server

Use the STATE parameter to specify the operational state of the server module. There are three possible states:

OFF	Prevents use of the module and clears all existing virtual circuits
ON	Allows normal use of the module
SHUT	Prevents use of the module for any new activity, but allows existing virtual circuits to complete their operation

The following command allows normal use of the module:

```
NCP>SET MODULE X25-SERVER STATE ON ...
```

The STATE parameter is optional and, by default, the state is ON.

For a complete list of states and their transitions, refer to the *VMS Network Control Program Manual.*

3.11 X.25 Access Module Commands

The X25–ACCESS module contains the database needed to connect the local host node to an X.25 connector node that can access specific PSDNs on behalf of the host nodes. Your local host node must be a DECnet–VAX node on which VAX PSI Access software is installed. The X.25 connector node may be a VMS node with VAX PSI software in multihost mode installed or an X25router node. Refer to Chapter 5 for an example of how to use NCP commands to configure the X.25 connector and host nodes.

Use the SET MODULE X25–ACCESS command to associate the name of the X.25 network you want to access with the name of the node serving as the connector to this network. You can optionally specify access control information for the link between your host node and the connector node.

3.11.1 Network Identification in an X.25 Access Module

Use the NETWORK qualifier with the SET MODULE X25–ACCESS command to identify the specific network you want to access through the X.25 connector node. The network name you specify must be the same as the one defined in the X.25 protocol module database, at the X.25 connector node.

The following command identifies the network PSS1 to which the local DECnet–VAX node with VAX PSI Access software wants access:

```
NCP>SET MODULE X25-ACCESS NETWORK PSS1 ...
```

You must specify the NETWORK qualifier and must associate with it the NODE parameter. You can optionally specify access control information to be used by VAX PSI Access software in connecting to the X.25 connector node.

3.11.2 X.25 Connector Node Identification

Use the NODE parameter to identify the node that is to provide connector or gateway services to the specified X.25 network. You must configure a DECnet–VAX node serving as an X.25 connector node with VAX PSI software in multihost mode. The connector node, also referred to as the target node, is the one with which the VAX PSI Access software on your local node establishes a DECnet link in order to transmit and receive X.25 and X.29 calls. The following command identifies the node ROBIN as the one that is to serve as the connector node to permit your local host node to access the network PSS1:

```
NCP>SET MODULE X25-ACCESS NETWORK PSS1 NODE ROBIN ...
```

ROBIN must be connected by an X.25 line to the PSDN. ROBIN must also contain an entry in its X.25 server module database indicating that the destinations of specific incoming X.25 calls are located on your host node (see Section 2.7).

3.11.3 Access Control Parameters in an X.25 Access Module

You have the option of specifying access control parameters to be used by the VAX PSI Access software on your local host node in establishing a link to the node serving as the X.25 connector. The access control parameters are the standard DECnet–VAX parameters: USER, PASSWORD, and ACCOUNT. The following command specifies the user identification PSI and password PSI the local node can use for an inbound connect when establishing a DECnet link with node ROBIN, the X.25 connector to the network PSS1:

```
NCP>SET MODULE X25-ACCESS NETWORK PSS1 NODE ROBIN -
_ USER PSI PASSWORD PSI
```

Refer to the connector node documentation to determine what use, if any, is made of these parameters.

3.12 Logging Commands

In order to log events, you must turn on logging. (The DECnet–VAX default has all events cleared.) To do so, use the SET LOGGING command. Use the same command to modify any of the logging parameters. To remove any or all parameters from the volatile database, use the CLEAR LOGGING command. You must turn the logging state to OFF before attempting to use the CLEAR LOGGING command.

Table 3–7 lists all logging parameters by function, and groups them according to operational categories.

Table 3–7 Logging Parameters and Their Functions

Parameter Function	Source-Related Parameter	Sink-Related Parameter
Identifies events	EVENTS event-list KNOWN EVENTS	
Identifies source for events	CIRCUIT circuit-id LINE line-id MODULE X25–ACCESS MODULE X25–PROTOCOL MODULE X25–SERVER MODULE X29–SERVER NODE node-id	
Determines location for logging events	SINK EXECUTOR SINK NODE node-id	
Assigns name to logging component		NAME sink-name
Sets state of logging component		STATE { HOLD | OFF | ON }

Source-related and sink-related parameters are mutually exclusive. Therefore, you cannot use parameters from both categories in a single command. Use the SET LOGGING EVENTS command to specify source-related events, and the SET LOGGING STATE command to specify sink-related events.

For a summary of event class and types and information about the specific events that the VMS operating system logs, see the *VMS Network Control Program Manual.*

The logging component is defined by the device or process that records the events released by the event logger. The logging component can be a LOGGING CONSOLE, LOGGING FILE, or LOGGING MONITOR. The LOGGING CONSOLE is a terminal or a file that receives events on the sink node in a format the user can read. A LOGGING FILE is a user-specified file on the sink node. The logging file component receives events in the standard DNA binary format. (Refer to the *DNA Phase IV Network Management Functional Specification* for a description of this format.) Instead of specifying the console and a file, you can specify a system- or user-supplied LOGGING MONITOR program to receive and process DNA format-specific events. This program could possibly receive event data and adapt user application network activity to reflect this data.

If the logging sink is the LOGGING MONITOR, DECnet–VAX uses the Operator Communication (OPCOM) facility to display formatted event messages on all terminals enabled as NETWORK (using REPLY /ENABLE=NETWORK). This generally includes the operator console (OPA0). The format of event messages OPCOM displays is similar to that used for console logging; however, because of restrictions in the size of messages that OPCOM can display, some messages may be truncated slightly, and node, circuit, and line counters are not displayed at all.

To identify the name of the logging component on the local node, use the NAME parameter. For example, if the component is a logging console file, the following command creates the file EVENTS.LOG into which formatted events will be logged:

```
NCP>SET LOGGING CONSOLE NAME SYS$MANAGER:EVENTS.LOG
```

To identify a logging monitor program as the logging component, use the NAME parameter followed by the program name. See Section 3.12.6.

Regardless of the logging component you use, parameter selection is the same. If you want to modify parameters for all logging on the network, then use the plural KNOWN LOGGING component when entering the SET LOGGING command.

Note: Because console logging uses normal VMS RMS file I/O, if a terminal is specified as a sink name, the terminal should not be used or allocated for any other purposes. For example, if you log in using such a terminal, events will be lost until you log out.

3.12.1 Event Identification

Events are defined by class and type. You can specify the kinds of events to be logged by using the following event-list format:

class.type

where:

class Identifies the DNA or system-specific layer to which the event pertains.

type Identifies a particular form of event, unique within an event class.

For example, to specify an event in the Routing layer, you use **event class** 4. The **event types** for this class range from 0 to 14. Event type 0 indicates aged packet loss, event type 1 indicates unreachable node packet loss, and so forth. Refer to the *VMS Network Control Program Manual* for a summary of events by class and type. Use the EVENTS parameter for the SET LOGGING command to specify those events to be logged. If you want to log all event classes and types, use the KNOWN EVENTS parameter. When defining the logging component, you must specify events to be logged.

When providing an event list for the EVENTS parameter, you can specify only one class for each instance of this parameter. However, several formats can define event types for a particular class. You can specify a single event type, a range of types, or a combination of the two. The following table illustrates these formats.

Event List	Meaning
4.4	Identifies event class 4, type 4
4.5–7	Identifies event class 4, types 5 through 7
4.5,7–9,11	Identifies event class 4, types 5, 7 through 9, and 11. Note that types must be specified in ascending order.

The following commands illustrate invalid event lists:

```
NCP>SET KNOWN LOGGING EVENTS 4.4,5.1          !INVALID COMMAND

NCP>SET KNOWN LOGGING EVENTS 4.7,3-4,1        !INVALID COMMAND
```

The first example specifies more than one event class. The second example specifies event types in numerically descending, rather than ascending, order.

You can use the asterisk (*) wildcard character in an event list. This character can replace only an event type. The following example illustrates the correct use of a wildcard character:

```
NCP>SET KNOWN LOGGING EVENTS 2.*
```

This command identifies all event types for class 2 events.

Two invalid uses of the wildcard character are as follows:

```
NCP>SET LOGGING FILE EVENTS *.2-5          !INVALID COMMAND
NCP>SET LOGGING FILE EVENTS 4.2-*          !INVALID COMMAND
```

The first command specifies specific event types for all classes, which is not allowed. Unless you use the KNOWN EVENTS parameter, you can specify event type information only for a single class. The second command uses a wildcard to specify a partial range of event types, also not allowed. The wildcard character denotes the entire range of event types for a given class.

3.12.2 Identifying the Source for Events

You can specify the particular source for which events apply, which can be either a node, a module, a circuit, or a line. For example, to monitor network activity for circuit DMC–0 connected to the local node, enter the following command:

```
NCP>SET LOGGING CONSOLE CIRCUIT DMC-0 . . .
```

Events that pertain to activity over this circuit are logged at the console by the event logger. You can perform the same operation for any remote node. If you specify no source for a component, the event logger logs events for all circuits, lines, modules, and nodes known to the local node or DTE.

Note that you can set only one source (a circuit, module, node, or line) as the source for events in a single command.

The command CLEAR LOGGING KNOWN EVENTS clears only events that are not associated with any specific source. To remove an event associated with a specific source, use the CLEAR LOGGING command that specifies that source.

3.12.3 Identifying the Location for Logging Events

You can log events either at the local node or a remote node. Use the SINK parameter to specify the location. For example, the following command routes all event information to the logging monitor program running on node DENVER:

```
NCP>SET LOGGING MONITOR SINK NODE DENVER . . .
```

If you do not specify a location, the local node is the default.

3.12.4 Controlling the Operational State of Logging

You can control the operational state of logging only for the local node. There are three logging states:

HOLD — Indicates that the sink is temporarily unavailable. Events destined for that location are queued.

OFF — Indicates that the sink is unavailable for receiving event information. Events are not logged for that sink.

ON — Indicates that the sink is available for receiving event information. This is the normal operational state.

Use the STATE parameter to specify the operational state of logging on the local node. The following command forces event information to be queued for all instances of the logging component on the local node:

```
NCP>SET KNOWN LOGGING STATE HOLD
```

Note that this control over logging does not affect the operational state of the node. Setting the default state to ON in the permanent database is recommended.

Note: You must specify the event logger object (number 26, name EVL) in the object database. If you experience difficulty with event logging, examine the event logger's own log file, SYS$MANAGER:EVL.LOG, for possible problems.

3.12.5 Event Logging Example

The example in this section illustrates how to use NCP event logging commands. You may want to log events normally to OPCOM for each node in the network. In addition, you may want each node to transmit its events to a single node to be stored in a file. For the three nodes—DENVER, TRNTO, and BOSTON—you could enter the following commands at each node to do this.

At nodes DENVER and BOSTON:

```
NCP>SET LOGGING MONITOR STATE ON
NCP>SET LOGGING MONITOR KNOWN EVENTS
NCP>SET LOGGING MONITOR SINK NODE TRNTO KNOWN EVENTS
```

At node TRNTO:

```
NCP>SET LOGGING MONITOR STATE ON
NCP>SET LOGGING MONITOR KNOWN EVENTS
NCP>SET LOGGING CONSOLE NAME SYS$MANAGER:NETEVENTS.LOG
NCP>SET LOGGING CONSOLE STATE ON
NCP>SET LOGGING CONSOLE KNOWN EVENTS
```

Events from all three nodes are logged to all terminals enabled as NETWORK (through the DCL command, REPLY/ENABLE=NETWORK) on node TRNTO. In addition, all local events are logged locally to the file NETEVENTS.LOG on node TRNTO. Note that the transmitting node always specifies the destination of the event logger output and causes its locally generated events to be sent to the receiving sink node to be logged.

3.12.6 Using a Logging Monitor Program

Instead of using a logging console or a logging file, you can specify a logging monitor program to receive and process events. The logging monitor is a system- or user-supplied program. The advantage of using a logging monitor program is that it can be tailored to the specific needs of the network manager.

You can write logging monitor programs in high-level languages and design them to perform specific functions desired by the network manager. Thus, the logging monitor program can be simple or complex, depending on its design and objective.

The following logging monitor example is a BASIC program called LOGGER.BAS. It records events released by the event logger and prints them to a terminal. Detailed information about the format of the events can be found in the *DNA Phase IV Network Management Functional Specification.*

```
10  ! TITLE LOGGER.BAS
    !
    ! This is a sample logging monitor program.
20    MAP (EVENT)                                          &
                BYTE    FUNCTION_CODE,                     &
                BYTE    SINK_FLAGS,                        &
                WORD    EVENT_CODE,                        &
                STRING  EVENT_TIME = 12,                   &
                WORD    SOURCE_NODE,                       &
                STRING  REST = 238

100 ! Record events released by the network event logger.
110  OPEN "SYS$NET" FOR INPUT AS FILE #1%, MAP EVENT
120  ON ERROR GOTO 998
200  ! Begin loop to extract events and write them to the terminal.
300  WHILE 1 = 1
400       GET #1%
410       EVENT_CLASS% = EVENT_CODE / 64%
420       EVENT_TYPE% = EVENT_CODE - 32% * (EVENT_CODE / 32%)
430       EVENT_CLASS$ = NUM1$ (EVENT_CLASS%)
440       EVENT_TYPE$ = NUM1$ (EVENT_TYPE%)
450       EVENT$ = EVENT_CLASS$ + "." + EVENT_TYPE$
460       PRINT "Event " ; EVENT$ ; " Reported"
499  NEXT
998  RESUME 999
999  END
```

To use a logging monitor, you must add the name of the program to the object database. For example, the following commands add the executable image LOGGER to the database and set LOGGER as the name of the logging monitor program:

```
NCP>SET OBJECT LOGGER NUMBER 0 FILE LOGGER.EXE
NCP>SET LOGGING MONITOR KNOWN EVENTS
NCP>SET LOGGING MONITOR STATE ON
NCP>SET LOGGING MONITOR NAME LOGGER
```

Sample output from the logging monitor program (LOGGER.EXE) is as follows:

```
Event 0.9 Reported
Event 0.9 Reported
Event 4.7 Reported
Event 4.10 Reported
Event 4.15 Reported
```

3.13 Network Access Control Commands

The system manager can specify NCP commands to provide for access control at the routing initialization level, at the system level during inbound logical link connections, and at the node level during inbound and outbound logical link connections. You can also use NCP commands to control proxy login access to individual accounts and network objects at the local node. The following sections indicate the NCP commands and parameters that you can specify for access control. Refer to Section 2.10 for a description of DECnet–VAX access control techniques.

3.13.1 Specifying Passwords for Routing Initialization

You can specify in your local configuration database transmit and receive passwords for each adjacent node. The transmit password is the one you send to the remote node and the receive password is the one you expect to receive from the remote node during the routing initialization sequence. Use the SET NODE command to specify these passwords. Each password can be one to eight alphanumeric characters in length. For example, the following command establishes transmit and receive passwords for the circuit or circuits connecting the local node with node TRNTO:

```
NCP>SET NODE TRNTO TRANSMIT PASSWORD VAX_NODE -
_ RECEIVE PASSWORD VAX_NODE
```

If the password contains one or more space characters, you must delimit it with quotation marks.

To remove transmit and receive passwords from the volatile database, use the CLEAR NODE command, as shown in the following example:

```
NCP>CLEAR NODE TRNTO RECEIVE PASSWORD TRANSMIT PASSWORD
```

To provide for increased security when a remote node requests a connection over a point-to-point circuit, you can use the circuit parameter VERIFICATION INBOUND to prevent your node from revealing its routing initialization password while requiring a password from the remote node.

When two nodes communicate over a point-to-point circuit, only one of the nodes can have the VERIFICATION INBOUND parameter set. The primary function of this parameter is to permit the system manager to restrict the nodes that can initialize over a particular circuit, especially over a dialup circuit.

When a dialup node attempts to establish a dynamic connection with your node, the dynamic asynchronous circuit entry is supplied automatically to your configuration database. This entry includes the circuit parameter VERIFICATION INBOUND, which prevents your node from supplying a password to the node requesting a dynamic connection, but requires a password from the node dialing in.

Note that if you specify VERIFICATION INBOUND for a circuit, you must also specify the node parameter INBOUND ROUTER or INBOUND ENDNODE, as appropriate, for the connecting node (see Section 3.13.3). This requirement applies to both dynamic and static asynchronous connections.

If, on the other hand, you are a user on a node with a terminal line (such as a VMS operating system running on a MicroVAX) and you expect to form a dynamic asynchronous connection with another node, you should specify a transmit password in your node database. For example, if you are at node WRKVAX and expect to form a dynamic connection with remote node VCLST1 on a VAXcluster, specify the following command to establish the transmit password for the dynamic circuit:

```
NCP>SET NODE VCLST1 TRANSMIT PASSWORD HOMENODE1
```

The remote node in a dynamic connection must specify the receive password it expects to receive from the local node. The system manager at remote node VCLST1 specifies the following command to indicate the password expected from node WRKVAX:

```
NCP>SET NODE WRKVAX RECEIVE PASSWORD HOMENODE1
```

3.13.2 System-Level Access Control Commands

You can use the SET NODE command to specify default privileged and nonprivileged access control strings for outbound logical link requests. Use the SET OBJECT command to specify privileges required to access certain objects during inbound logical link requests. You can also use the SET OBJECT command to specify a default access control string. For NCP commands to be executed at remote nodes, you can either supply explicit access control information in the node specification, as parameters in the command, or by default.

3.13.2.1 Establishing Default Privileged and Nonprivileged Accounts

Use the SET NODE command to specify default access control information for connecting to remote nodes. If you have not specified explicit access control information in an outbound logical link request, this default information is sent with the request. For example, the following command specifies both privileged and nonprivileged user names and passwords for node DENVER:

```
NCP>SET NODE DENVER -
_ NONPRIVILEGED USER NETNONPRIV PASSWORD NONPRIV-
_ PRIVILEGED USER NETPRIV PASSWORD PRIV
```

You should specify default information for all remote nodes with which you want to have the option of using default access control.

3.13.2.2 Specifying Privileges for Objects

Use the SET OBJECT command with the PRIVILEGE parameter to specify those privileges that cause the privileged user account to be used rather than the nonprivileged user account. The privilege list accompanying the parameter specifies those privileges required for all inbound connections to that object. For example, you may want to make the FAL object accessible to any network user, whereas you want to control access to the NML object. The following command specifies privileges for the NML object in this instance:

```
NCP>SET OBJECT NML PRIVILEGES OPER
```

You need not specify privileges for FAL because it requires only NETMBX and TMPMBX privileges.

3.13.2.3 Setting Default Inbound Access Control Information

Use the SET OBJECT command with the USER, ACCOUNT, and PASSWORD parameters to specify default inbound access control information. For example, the following command specifies default information that the local DECnet–VAX node can use for inbound connects from SLD:

```
NCP>SET OBJECT HLD USER NETNONPRIV PASSWORD NONPRIV
```

3.13.2.4 Indicating Access Controls for Remote Command Execution

You use access control for remote NCP command execution. When you enter the SET EXECUTOR NODE and TELL commands, you can explicitly specify access control information, or you can default to information contained in the configuration database.

Two formats exist to supply access control information explicitly for these commands. You can use either a standard VMS node specification *node"user password account"*:: or the NCP parameter USER, ACCOUNT, or PASSWORD. For example, the following commands perform the same operation:

```
NCP>SET EXECUTOR NODE TRNTO"GRAY MARY"::

NCP>SET EXECUTOR NODE TRNTO USER GRAY PASSWORD MARY
```

The same formats exist for the TELL command. Use of the standard VMS node specification format allows you to use a logical name as the node-id for these commands. It is possible to override access control in a logical name with explicit access control information in the command.

You can also use access control information to cause NML to run under an account other than the default DECnet privileged account on the local node. Enter the following command for this purpose:

```
NCP>SET EXECUTOR NODE TRNTO"user-id password"
```

3.13.3 Node-Level Access Control Commands

At the node level, you can specify access control commands that determine what connections can be made. If your node expects to receive dialup dynamic asynchronous connection requests, you can check the type of the dialup node before permitting the connection.

The NCP commands SET NODE ACCESS and SET EXECUTOR DEFAULT ACCESS, when used together, allow you to partition your network to allow specific access for each node. For example, assume that there are 10 nodes in your network, named A through J. The executor is node A. Because most network traffic occurs among nodes A, B, and C, you could use the following commands to allow unrestricted incoming and outgoing logical link connections among those nodes:

```
NCP>SET NODE A ACCESS BOTH

NCP>SET NODE B ACCESS BOTH

NCP>SET NODE C ACCESS BOTH
```

Next, assume that you want to allow local users to initiate connections to node D, but restrict connections from that node. Enter the following command:

```
NCP>SET NODE D ACCESS OUTGOING
```

Finally, assume that you want to allow incoming logical link connections from all other remote nodes (E through J), but restrict outgoing connections from the executor node. Enter the following command:

```
NCP>SET EXECUTOR DEFAULT ACCESS INCOMING
```

Note: The executor checks for a node ACCESS entry before it checks for the DEFAULT ACCESS entry. Remember that, if the executor's state is set to OFF or SHUT, no logical links are allowed.

You can indicate the type of node that can connect to your node over a point-to-point circuit by specifying the INBOUND parameter with the SET NODE command. The INBOUND parameter enables you to check the type of a connecting node before you form a dynamic connection with the node. For example, if you expect the VMS node WRKVAX to initiate a dynamic connection by dialing in to your node over a specific terminal line, you can specify the following in your node database:

```
NCP>SET NODE WRKVAX INBOUND ENDNODE
```

If the node WRKVAX dials in as a router, rather than as an end node, the dynamic connection is not formed. If you specify INBOUND ROUTER for the node and it dials in as an end node, the dynamic connection is permitted.

Note that when you specify the node parameter INBOUND, you must also set the circuit parameter VERIFICATION INBOUND for the circuit over which the connection is to be made (see Section 3.13.1). If you do not set VERIFICATION INBOUND for the circuit, the node parameter INBOUND is ignored.

3.13.4 Proxy Login Access Control Commands

You can control proxy login access for accounts by modifying the executor database. To control proxy login for network objects, modify the object database.

Access to individual accounts on the local node by proxy login is enabled by the INCOMING PROXY and OUTGOING PROXY settings in the executor database. The default values for these parameters permit both incoming and outgoing proxy access. The default setting is the recommended option. You can, however, use the SET EXECUTOR command to modify the INCOMING PROXY and OUTGOING PROXY values at the local node.

The default value of the INCOMING PROXY and OUTGOING PROXY entries in the executor database are equivalent to entering the following commands:

```
NCP>SET EXECUTOR INCOMING PROXY ENABLED
NCP>SET EXECUTOR OUTGOING PROXY ENABLED
```

The system manager has the option of changing the default values for proxy login. The following examples establish that any proxy login to or from the local node is prohibited:

```
NCP>SET EXECUTOR INCOMING PROXY DISABLED
NCP>SET EXECUTOR OUTGOING PROXY DISABLED
```

Note that if proxy access has been enabled for specific network objects, the previous SET EXECUTOR commands would not prevent a user from using a proxy account. Proxy access for network objects must also be explicitly disabled. The proxy access characteristics established in the object database take preference over the proxy access characteristics established in the executor database.

To display the value of the proxy entries for your node, enter the following command:

```
NCP>SHOW EXECUTOR CHARACTERISTICS
```

If proxy login access is enabled at your node, the resultant display includes the following:

```
Incoming Proxy            = Enabled
Outgoing Proxy            = Enabled
```

When incoming proxy login access is enabled, the remote user can access a file accessible to the local account to which he has default proxy access by using the node specification NODE:: in the standard VMS file specification. For example, a remote user can specify the following form of file specification to access a file on an account on node TRNTO to which he has default proxy access:

```
TRNTO::filename
```

In the following example, the remote user requests access to the local account PROXY_N, assuming proxy access is allowed:

```
TRNTO"PROXY_N"::filename
```

In this example, PROXY_N may be the default proxy account, or it may be another proxy account established for the remote user.

To override proxy login, the remote user with a proxy account on a node can specify NODE"":: in the file specification, causing the default nonprivileged DECnet account to be used, because explicit null access control is passed to the remote node.

The SET EXECUTOR command grants proxy login access to specific accounts. Similarly, you can permit or deny proxy login access to specific network objects, by using the SET OBJECT command to modify the object database. Access to a network object through a proxy account is controlled by the PROXY parameter in the object database. By default, DECnet–VAX has set in the configuration database PROXY values for some network objects. These default values are the recommended values. To specify or modify the PROXY parameter for an object, use the SET OBJECT command with the PROXY parameter. In the following example, the outgoing proxy access option is set for the object FAL:

```
NCP>SET OBJECT FAL PROXY OUTGOING
```

To display the setting for the PROXY parameter in the database, use the SHOW OBJECT command with the CHARACTERISTICS parameter, as in the following command:

```
NCP>SHOW KNOWN OBJECT CHARACTERISTICS
```

The resulting display lists the database entries for each known object, indicating any proxy access that is enabled for the object. For object MAIL, the display is as follows:

```
OBJECT = MAIL

Number                    = 27
User id                   = NETNONPRIV
Password                  = TREWQ
Proxy access              = outgoing
```

System managers use the Authorize Utility to manage the permanent proxy database, NETPROXY.DAT. Information in NETPROXY.DAT is used to construct a volatile database in the NETACP process when DECnet is started up. An NCP command, SET KNOWN PROXIES ALL, updates the volatile proxy database if changes are made while the network is running. This command clears the contents of the volatile proxy database and rebuilds it from the permanent proxy database. SET KNOWN PROXIES ALL is executed as part of the SYS$MANAGER:STARTNET command procedure.

While SET KNOWN PROXIES ALL updates the volatile proxy database, all modifications of the permanent proxy database are handled by means of the Authorize Utility. You may not modify the individual entries in the volatile database.

3.14 Monitoring the Network

You can monitor network activity in one of two ways: by using the NCP command SHOW or by using the event logging facility and the SET LOGGING command. This section discusses the use of the SHOW and LIST commands. Refer to Section 2.9 for a discussion of events and event logging, and Section 3.12 for a description of the SET LOGGING command.

NCP provides commands to display information about network components, whether they are defined in the volatile or permanent database. The NCP command SHOW displays information about components for the running network. The NCP command LIST performs a similar function, except that it lets you display and verify information in the permanent database. In many cases, this information is a subset of the information displayed for the volatile database.

In general, the SHOW command allows you to monitor the operation of the running network. For example, whenever someone changes the state of a circuit, the configuration of the running network in terms of reachable and unreachable nodes may be changed as well. A circuit failure could have the same effect. NCP allows you to display the status of network circuits, lines, modules, and nodes, and thereby to detect such conditions.

When you enter the SHOW and LIST commands, NCP allows you to select components and display types. You can choose among several display types, depending on the information you want. The display type determines the format and type of information NCP displays. Display types are described in the following table.

CHARACTERISTICS	Includes static information that is usually specified in the configuration database. Depending on the component, this information may include the identification of a local node and relevant routing parameters, the names and numbers of known network objects, and the identification and cost of circuits connected to the local node. For VAX PSI, the information may include identification of a local DTE and relevant parameters; packet size, window size, and other network parameter values; device identification; and timer values.
STATUS	Includes dynamic information that usually reflects network operations for the running network. Depending on the component, this information may include the local node and its operational state, reachable and unreachable nodes and their operational states, and circuits with their operational states. For VAX PSI, the information reflects the operation of the running VAX PSI software. Depending on the component, this may include identification of the line with its operational state.
SUMMARY	Includes only the most useful information derived from both static and dynamic sources. This information is usually an abbreviated list of information provided for both the CHARACTERISTICS and STATUS display types. For VAX PSI, it is usually an abbreviated list of information provided for the STATUS display type.
EVENTS	Includes information about events currently being logged for the logging component. This display type is valid only for the SHOW LOGGING and LIST LOGGING commands.
COUNTERS	Provides counter information for circuits, lines, modules, and nodes, including the local node. Counters are discussed in the parts of this section that describe the circuit, line, module, and node commands.

If you do not specify a display type when entering a SHOW or LIST command, SUMMARY is the default. Examples of these display types and their formats are given in the *VMS Network Control Program Manual*.

When you display information about network components, you can use either the singular or plural form of the component, as shown in the following example:

```
NCP>SHOW NODE BOSTON CHARACTERISTICS
                .
                .
                .

NCP>SHOW KNOWN NODES CHARACTERISTICS
                .
                .
                .
```

For several components, there is a second form of the plural. This form is the word ACTIVE. Whereas the word KNOWN displays information for components available to the local node, the word ACTIVE displays information for all active components—that is, components whose state is other than OFF.

Use the word ACTIVE with circuit, line, node, and logging components. For example, the following command displays the characteristics for all active nodes in the network:

```
NCP>SHOW ACTIVE NODES CHARACTERISTICS
                .
                .
                .
```

The word ADJACENT is also used as a plural in the SHOW NODE command, as in the following example:

```
NCP>SHOW ADJACENT NODES STATUS
```

All NCP display commands optionally allow you to direct the information displayed to a user-specified output file. For example:

```
NCP>SHOW KNOWN LOGGING SUMMARY TO SYS$MANAGER:NET.LOG
```

This command creates the file SYS$MANAGER:NET.LOG containing summary information of all known logging for the running network. The default file type is LIS. If the specified file already exists, NCP appends the display information to that file. If you do not specify an output file, SYS$OUTPUT is the default.

NML must have the BYPASS privilege to display passwords for the SHOW or LIST command; if it does not, no information appears if you use SHOW, and the "no access rights" message appears when you use LIST.

4 DECnet–VAX Host Services

DECnet–VAX can act as the host node in performing the following services for unattended systems:

- Downline loading of an unattended system: transferring a copy of an operating system file image from a VMS node to a target node.
- Downline loading of a satellite node in a Local Area VAXcluster from a VMS node.
- Downline loading of various servers from a VMS node.
- Downline loading of an RSX–11S task from a VMS node.
- Upline dumping of memory from an unattended system: transferring a copy of a memory image from an unattended target node to your VMS node.
- Connecting to a remote console: permitting a VMS terminal to act as the console for certain unattended systems, such as the DIGITAL Ethernet Communications Server running Router Server Software.

This chapter describes these operations. Note that host services are not available over asynchronous lines.

4.1 Loading Unattended Systems Downline

DECnet–VAX allows you to load an unattended system using the services provided by the Maintenance Operations Module (MOM). MOM provides a set of maintenance operations over various types of circuit by using the Maintenance Operations Protocol (MOP). Downline loading involves transferring a copy of the file image of a remote node's operating system from a VMS node to the unattended target node. For example, DECnet–VAX permits you to load an RSX–11S operating system file image from your VMS node downline to a target node. Downline loading can be initiated by a VMS operator or by the target node. Both procedures are discussed in this section.

To understand downline loading, it helps to distinguish the nodes involved in the loading sequence. In the following node descriptions, the command node and the executor node can be the same or different nodes, but cannot be the target node.

- **Command node.** An operator-initiated downline load request originates at the command node. You use the NCP command LOAD or TRIGGER to initiate this request.
- **Executor node.** The executor node actually performs a downline load or trigger operation.
- **Target node.** The target node receives the bootstrap loaders and the system image file.

4.1.1 Downline System Load Operation

Downline loading is initiated in one of two ways:

- An operator initiates the operation with the NCP command LOAD or TRIGGER. This is called the operator-initated mode.
- The target node initiates the operation by triggering its bootstrap ROM and sending a program load request to one or more potential executor node. This is called the target-initated mode.

The operator-initiated mode is used to service maintenance operations generally requested by an interactive operator. The operator enters maintenance requests using NCP, which delivers the request to the Network Management Listener (NML), which, in turn, spawns a process to execute the MOM image. This process then acts upon the request. With an operator-initiated load, the local node starts the operation by sending a trigger message to the target node. Essentially, the trigger message tells an unattended target node to reboot. After the target node is triggered, it loads itself in whatever manner its primary loader is programmed to operate. The target node can request a downline load from either the executor that just triggered it or from another adjacent node. The target node can also load itself from its own mass storage device.

The target-initiated mode is used to service unsolicited maintenance requests. In this mode, NETACP on the host system listens to circuits with service enabled for any MOP request directed to the local node or to a multicast address. When such a request arrives, NETACP creates a process to execute the MOM image. This process reads the request and processes it.

There are some subtle differences in the two modes in which MOM can execute. In target-initiated mode, MOM runs in a process created by NETACP. The information that MOM has for fulfilling the unsolicited request comes firstly from the request itself. This can be data such as the "software identification" requested by the node and the type of communication device that the requesting node is using to make the request. Additional information required to fulfill the request can be obtained from the volatile database on the local node. The information supplied in the MOP request takes precedence over the information in the volatile database.

In operator-initiated mode, MOM runs in a subprocess spawned by NML. Information for fulfilling a request can come from NCP parameters supplied by the operator, or from the volatile database. Information supplied in the command line takes precedence over information obtained from the volatile database.

4.1.1.1 Target-Initiated Downline Load

In a target-initiated downline load, the target node sends a program load request message. This message is a request for any eligible node to perform the load. The program load message can potentially specify a number of fields, including a software identification, a software type, and a service device.

If the load is target-initiated and the circuit has SERVICE enabled, NETACP detects a packet of MOP messages coming over the circuit and sets the circuit's substate to service mode. NETACP then verifies whether MOM can obtain a file specification before it starts the MOM process. If a software identification is provided in the MOP request, or if the request is for a secondary or tertiary loader, NETACP starts the MOM process. Otherwise, NETACP searches the node database for an entry that matches the hardware address. If there is no entry, NETACP drops the request. If an entry is found, NETACP starts the MOM process. NETACP creates a MOM process named *MOM_circuit-id_process-number* (for example, MOM_UNA-0_1) to perform the load. The MOM process then attempts to perform the load. If the circuit between the nodes is a point-to-point circuit, MOM searches the node database for the information required to service the request. The node entry is selected based on matching the service circuit parameter. (The service circuit is the one used for loading,)

If the connecting circuit is an Ethernet, MOM determines if the request was directed to the multicast address or to the local node. If the request was directed to the local node, MOM follows the same process as for point-to-point circuits. If the request was for the multicast address, MOM volunteers to perform the load. If a software identification is provided in the request, or if a node entry is found that matches the hardware address of the remote node, MOM sends a message to the requesting node volunteering to perform the load. If MOM does not get a response from the remote node, it drops the received packet and exits. Otherwise, it services the remainder of the load.

When the MOM process has the required information to perform the load (in the case of a volunteer, MOM has also received a response), it performs the load operation. Figure 4–1 illustrates this loading process.

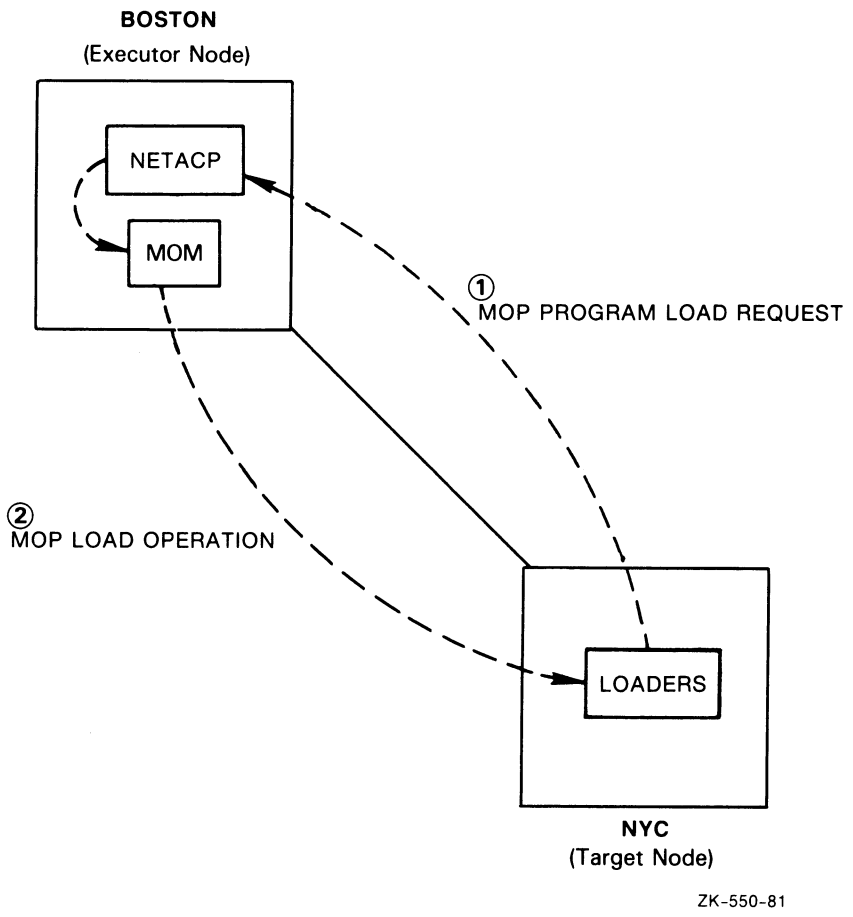
If the target node supplies a software identification along with its MOP program load request, then this identification is the load file specification. If no software identification is supplied, then the program type field is consulted to determine which kind of file the target node is requesting.

There are four possible values for program type:

- Secondary loader
- Tertiary loader
- Operating system
- Management file

The secondary loader, tertiary loader, and operating system files designate image specifications, while the management file value designates a general data specification. You can use the management file to specify additional information for downline loading that certain systems may need.

Figure 4–1 Target-Initiated Downline Load



After it knows the type of file being requested, NETACP can obtain a file specification in one of three locations. The first location searched is the program load request message that MOM received. The second is the node database entry for the requesting node. If the file specification is not in the node entry or in the program request, and the request is for an operating system or a management file, the MOM process aborts the service request and exits. If the request is for a secondary or tertiary loader, MOM attempts to concatenate the file specification. The components it uses for building the file specification are SEC or TER (for secondary and tertiary), the mnemonic of the service device on the requesting node, and a file type SYS. For example, if the remote node has a UNA as its service device, and is requesting the secondary loader, MOM creates the file specification SECUNA.SYS. MOM checks for the existence of the file before volunteering to perform the load. If the file is not available, MOM aborts the service request and exits.

The typical load sequence for a target-initiated request is as follows. The first program to run at the target node is the primary loader. Typically, this program is either executed directly from the target node's bootstrap ROM, or it is in the microcode of the load device. After the target node's primary

loader is triggered, the target node sends a message requesting a program load to any eligible executor node. (The executor node may be a specific node defined by the target node, or any node on the Ethernet.) Usually, the primary loader requests a secondary loader program, which then may request a tertiary loader, which, in turn, may request an operating system. The final module to be loaded is the management file. In this sequence, each program or file requests the next one until the management file is loaded.

The secondary loader is small and is always sent as a single message. The tertiary loader, operating system, and management file are larger and are sent in segments. For all of these, the last segment is followed by a PARAMETER LOAD WITH TRANSFER ADDRESS message, which supplies the start address of the image just loaded and contains extra values for the node identification and the host identification.

The downline load sequence varies when a request originates from a node in a Local Area VAXcluster. A node in a Local Area VAXcluster may need more input parameters than are currently defined in the volatile node database. Thus, you need to be able to dynamically configure the VMS image to be loaded, and to transfer more parameters to the target system than those accommodated by the Parameter Load and Transfer packet. To accommodate this need, MOM calls on the services of a **load assist agent** to help fulfill a downline load request. The load assist agent is an image that makes calls back to MOM with data that describes the image to be loaded on the target node.

For downline loads to nodes in Local Area VAXclusters, MOM delivers all load requests to load assist agents. The node parameter LOAD ASSIST AGENT identifies a specific agent by file name. Section 4.1.2.7 describes the procedure for specifying the LOAD ASSIST AGENT file specification. Another node parameter, LOAD ASSIST PARAMETER, passes an individual value to a load assist agent file. Section 4.1.2.8 describes the procedure for specifying the LOAD ASSIST PARAMETER value.

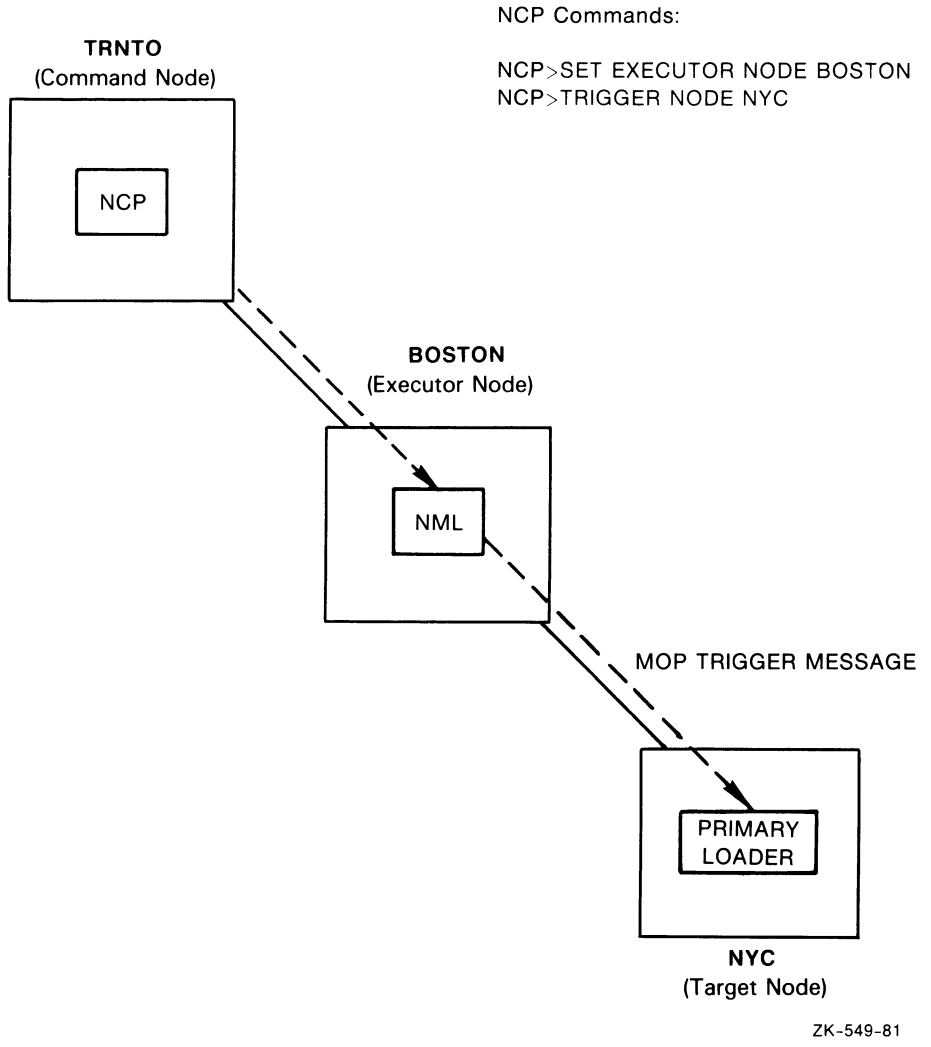
4.1.1.2 Operator-Initiated Downline Load

An operator-initiated load uses NCP to directly request MOM to perform the load operation. The target node's primary bootstrap may or may not have to be triggered depending on the state of the target. The target node is triggered primarily to put it into a known state and to force it to supply program request information. Figure 4–2 illustrates the loading process.

Use the NCP command LOAD or TRIGGER to perform an operator-initiated downline load. The TRIGGER command allows you to directly trigger the remote node's bootstrap ROM, which causes the target node to send its host a request for a load operation. The programs to be loaded may come from a local disk file on the target node, another adjacent node, or the command node.

Note that the TRIGGER command may or may not initiate a downline load. One of the functions of this command is to simulate the operation that occurs when you push the BOOT button on the target node. A bootstrap operation from the local disk may result.

Figure 4–2 Operator-Initiated Downline Load



When you use the LOAD command, the executor node proceeds with the load operation according to the options specified in the initial load request. You obtain any required information that has been defaulted from the volatile database. With this information, the executor is thereby able to control the load sequence.

Section 4.1.2 describes the TRIGGER and LOAD commands, their parameters, and examples of their use.

4.1.1.3 Load Requirements

Prior to attempting a downline load operation, you must ensure that nodes, lines, and circuits meet the following requirements:

- The target node must be connected directly to the executor node. The executor node provides the line- and circuit-level access.
- The primary loader must either be a cooperating program in the target or in the microcode of the target's device. The downline load operation usually involves loading a series of bootstraps, each of which requests the next program until the operating system itself is loaded.
- The executor must have access to the load files. The location of the files can be either specified in the load request or defaulted to in the volatile database. Remote files are obtained through remote file access operations. (Refer to the examples in Section 4.1.2.4.)
- The target node must be able to recognize the trigger operation or must be triggered manually.
- The circuit involved in the load operation must be enabled to perform service functions. It must also be in the ON or SERVICE state; a multiaccess Ethernet circuit must be in the ON state. For example, the following command readies circuit DMC-0 for downline loading node BANGOR in the network example:

```
NCP>SET CIRCUIT DMC-0 SERVICE ENABLED STATE ON
```

- You must turn the line to ON and specify a service timer for the line. This timer sets the MOP timeout constant for retransmission if necessary. This is the method by which MOP handles error recovery. The default for the service timer is 4000 milliseconds. In the following example, the command sets the retransmission frequency to 5000 milliseconds and turns on line DMC-0:

```
NCP>SET LINE DMC-0 SERVICE TIMER 5000 STATE ON
```

Refer to the *Maintenance Operation Protocol Functional Specification* for a complete description of MOP error recovery.

4.1.2 Downline Load Parameters

The most convenient method of downline loading involves setting default information in the volatile database. The operator can use the NCP command SET NODE to establish default information for the target node in the volatile database. These default parameters are also used for target-initiated downline loads, though the MOP program load message can override some of the defaults. (This default method is discussed later in this chapter.) Alternatively, you can override the default by specifying several parameters for the NCP command TRIGGER or LOAD. The following sections describe each parameter and illustrate their use.

4.1.2.1 TRIGGER Command

The TRIGGER command triggers the bootstrap mechanism of a target node, which causes the node to request a downline load. Because the system being booted is not necessarily a fully functional network node, the operation must be performed over a specific circuit. To bring up the system at the target node, use either the TRIGGER NODE or TRIGGER VIA command. If you use the TRIGGER NODE command and do not specify a loading circuit, the executor node obtains the circuit identification associated with the target node from its volatile database. If you use the TRIGGER VIA command, which indicates the loading circuit but not the node identification, the executor node uses the default target node identification in its volatile database. To identify the target node in the volatile database, specify the SET NODE command with the appropriate SERVICE CIRCUIT parameter, which establishes the circuit to be used for loading.

The following command triggers node BANGOR in the network example:

```
NCP>TRIGGER NODE BANGOR VIA DMC-0
```

Note that this command specifies a DDCMP circuit over which the operation is to take place. Figure 4–3 also illustrates the use of the TRIGGER command for downline loading over a DDCMP circuit in the network example.

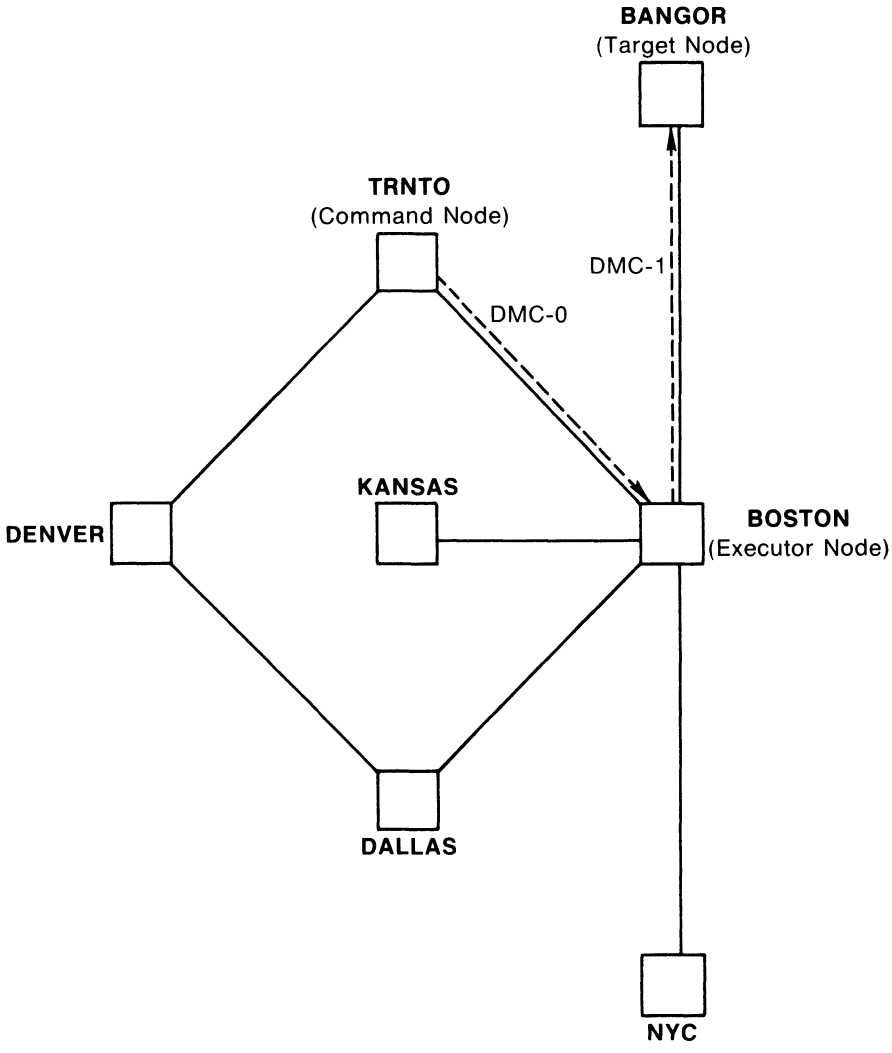
If downline loading is to occur over an Ethernet circuit, the executor node uses the Ethernet physical address of the target node to distinguish it from other adjacent nodes on the same Ethernet circuit. The PHYSICAL ADDRESS parameter for the target node is required in the TRIGGER VIA command and optional in the TRIGGER NODE command.

If you do not specify an Ethernet physical address in the TRIGGER NODE command, DECnet–VAX derives one from the target's node number and attempts to trigger the node. A target node that is running DECnet software has set its own Ethernet physical address and recognizes the address; otherwise, the target node recognizes only the Ethernet hardware address set by the manufacturer. If unsuccessful in triggering the node, DECnet–VAX attempts to use the Ethernet hardware address of the target node from the volatile database to trigger it. You can set in the volatile database the Ethernet hardware address originally assigned to the target node's DEUNA or DEQNA controller by specifying the HARDWARE ADDRESS parameter in the SET NODE command. (Refer to Section 3.3.4 for a description of Ethernet addressing.)

Figure 4–4 illustrates the use of the TRIGGER NODE command for downline loading a target node over Ethernet circuit UNA-0.

When you use the TRIGGER command, how the system load is performed may not always be obvious. Essentially, this command provides the trigger message that controls the restart capability for an unattended target node. After the target node is triggered, it loads itself in whatever manner its primary loader is programmed to operate. The target node can request a downline load from either the executor that just triggered it or another adjacent node, or the target node can load itself from its own mass storage device.

Figure 4–3 Operator-Initiated Downline Load over DDCMP Circuit (TRIGGER Command)



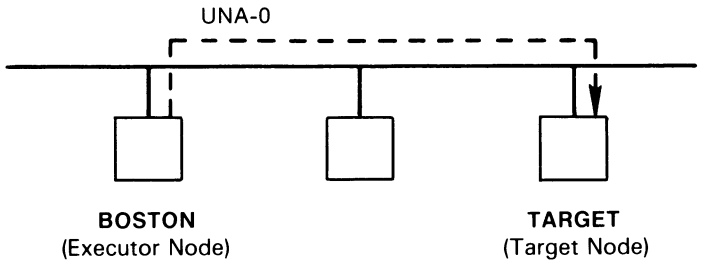
NCP Commands:

```
NCP>SET EXECUTOR NODE BOSTON
NCP>SET LINE DMC-1 STATE ON
NCP>SET CIRCUIT DMC-1 STATE ON
NCP>TRIGGER NODE BANGOR SERVICE PASSWORD FEFEFEFE
```

ZK 1867 84

One parameter that you can specify for the TRIGGER command is SERVICE PASSWORD. This parameter supplies a boot password, which may be required by the target node (see Section 4.1.2.12). If you do not specify this parameter, a default value from the volatile database is used. Use the SET NODE command to establish a default value for this parameter in the volatile database. If no value is set in the volatile database, the value is 0.

Figure 4–4 Operator-Initiated Downline Load over Ethernet Circuit (TRIGGER Command)



NCP Commands:

```
NCP>SET LINE UNA-0 STATE ON
NCP>SET CIRCUIT UNA-0 STATE ON
NCP>TRIGGER NODE TARGET PHYSICAL ADDRESS AA-00-04-00-CB-04 VIA UNA-0
```

ZK-1213-82

4.1.2.2 LOAD Command

Use the LOAD NODE and LOAD VIA commands to load software downline to a target node. For example, the following command loads node BANGOR:

```
NCP>LOAD NODE BANGOR
```

The LOAD NODE command requires the identification of the service circuit over which to perform the load operation. If you do not explicitly specify a service circuit in this command, the executor node uses the SERVICE CIRCUIT from the volatile database entry for the target node. You must use the SET NODE command to include the SERVICE CIRCUIT entry in the volatile database. Alternatively, you can explicitly include the circuit in the command LOAD NODE BANGOR VIA DMC–0.

You could also use the LOAD VIA command to specify the circuit over which to perform a downline load. For example, to load using circuit DMC–0 connected to the executor node, enter the following command:

```
NCP>LOAD VIA DMC-0
```

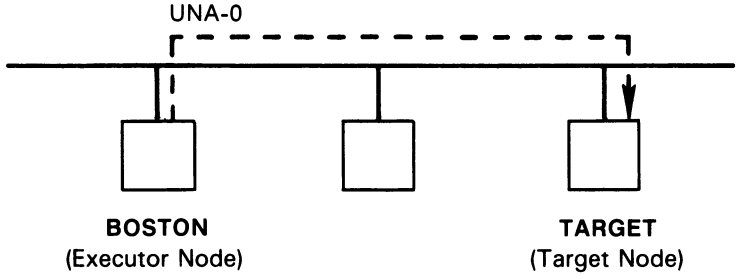
The executor node obtains the rest of the necessary information from its volatile database. The LOAD NODE and LOAD VIA commands work only if the target node can be triggered by the executor or if the target has been triggered locally.

If the loading circuit is a multiaccess Ethernet circuit, the executor node uses the Ethernet physical address of the target node to differentiate the node from other adjacent nodes on the same Ethernet. You must specify the PHYSICAL ADDRESS parameter in the LOAD VIA command, which does not identify the target node, but is optional in the LOAD NODE command.

If you do not specify the PHYSICAL ADDRESS parameter in the LOAD NODE command, DECnet–VAX derives the Ethernet physical address from the target node number and attempts to load the target node. A target node running DECnet software has set its own Ethernet physical address and recognizes this address; otherwise, the target node recognizes only the Ethernet hardware address set by the manufacturer. If unsuccessful in loading the node, the executor node attempts the load using the Ethernet hardware address of the target node from the volatile database. You can set in the volatile database the Ethernet hardware address originally assigned to the target node's DEUNA or DEQNA controller, by specifying the HARDWARE ADDRESS parameter in the SET NODE command. (Refer to Section 3.3.4 for a description of Ethernet addressing.)

Figure 4–5 illustrates how to use the LOAD command for downline loading over Ethernet circuit UNA–0.

Figure 4–5 Operator-Initiated Downline Load over Ethernet Circuit (LOAD Command)



NCP Commands:

```
NCP>SET NODE TARGET SERVICE CIRCUIT UNA-0
NCP>SET LINE UNA-0 STATE ON
NCP>SET CIRCUIT UNA-0 STATE ON
NCP>LOAD NODE TARGET PHYSICAL ADDRESS AA-00-04-00-CB-04
```

ZK-1214-82

If you choose to override the default parameters for the LOAD commands, you can control the following aspects of the load sequence:

- The host node that the target node is to use when the target comes up

  HOST node-id

- The identification of the load file

  FROM file-id

- The identification of the loader programs

  SECONDARY LOADER file-id
  TERTIARY LOADER file-id

- The software type to be loaded downline first

  SOFTWARE TYPE software-type

  where:

  software-type can be any of the following:

  SECONDARY LOADER
  TERTIARY LOADER
  SYSTEM
  MANAGEMENT FILE

  If you do not specify SOFTWARE TYPE in the first MOP program request, the NCP command, or the volatile database, the default is SECONDARY LOADER.

- The identification of the CPU type and the corresponding software identification

  CPU cpu-type
  SOFTWARE IDENTIFICATION software-id

- The identification of the target node's line device type that is to handle service operations

  SERVICE DEVICE device-type

- The identification of the service password for triggering the target node's bootstrap mechanism

  SERVICE PASSWORD hex-password

- The identification of the VMS image that defines system software for downline loading to a node in a Local Area VAXcluster

  LOAD ASSIST AGENT file-spec

- The identification of an additional parameter to be included in a load assist agent file

  LOAD ASSIST PARAMETER item

When entering the LOAD NODE and LOAD VIA commands, you can specify any or all of the preceding parameters. Any parameter not specified in the command defaults to whatever information is specified in the volatile database. Use the SET NODE command to establish default information for the target node's parameters in the volatile database.

4.1.2.3 Host Identification

At the end of the load sequence, the target receives a message with the name of the host and places that name in its volatile database. The target can then use the HOST node-id for downline task loading applications. The host may be the executor node or any other reachable node except for the target itself. Use the SET NODE command to specify a default host node where the target will find the files used to load tasks downline. For example, the following command sets the host to node NYC when node BANGOR comes up as a network node (if BANGOR has the necessary DECnet software):

```
NCP>SET NODE BANGOR HOST NYC
```

If you do not specify the host, the executor serves as the default host. You can override any default information by using the HOST parameter for the LOAD command.

4.1.2.4 Load File Identification

The load files are the files to be loaded downline to the target node. These files include the secondary loader, the tertiary loader and the operating system image, and the file specification for the management file. You can specify default load file names in the volatile database with the SET NODE command. Load file specifications default to MOM$SYSTEM:filename.SYS, where *filename* is usually provided by the volatile database. If the files are on another node, specify *node-id::* at the beginning of the file specification.

Figure 4–6 illustrates how to use the LOAD NODE command for loading a target node over a DDCMP circuit.

If you do not include the secondary and tertiary file names in the LOAD command or as entries in the volatile database for the target node, the load files are selected according to the service device type on the target system, not by the device type on the executor. The default secondary and tertiary loader files are listed in Table 4–1. The DECnet–11S kit includes the files listed in Table 4–1.

If you do not specify load file names in the target's load request, the NCP command LOAD, or the volatile database, NML provides them by concatenating the service device with the prefix SEC or TER. For example, if the service device is a DMC, NML uses the file names SECDMC.SYS and TERDMC.SYS.

If, however, you include the secondary and tertiary file names as entries in the volatile database for the target node, they can override the default loader files shown in Table 4–1. By using the SET NODE command, you can select your own special load files for a particular target node. If you do not specify the load files, you can change the service device type at the target node without changing the target node's database entry at the executor node.

The system image file entry in the host node's volatile database serves as the default file name for the operating system to be downline loaded. This file name is required when the target node is to be loaded, but it can be supplied by the LOAD command.

Table 4–1 Default Loader Files by Target Device Type

Device Type	Secondary Loader	Tertiary Loader
DA	SECDA.SYS	TERDA.SYS
DEBNA	SECBNA.SYS	TERBNA.SYS
DELQA	SECLQA.SYS	TERLQA.SYS
DELUA	SECLUA.SYS	TERLUA.SYS
DEQNA	SECQNA.SYS	TERQNA.SYS
DESVA	SECSVA.SYS	TERSVA.SYS
DL11	SECDL.SYS	TERDL.SYS
DMB32	SECDMB.SYS	TERDMB.SYS
DMC11	SECDMC.SYS	TERDMC.SYS
DMP11	SECDMP.SYS	TERDMP.SYS
DMV11	SECDMV.SYS	TERDMV.SYS
DP11	SECDP.SYS	TERDP.SYS
DPV11	SECDPV.SYS	TERDPV.SYS
DQ11	SECDQ.SYS	TERDQ.SYS
DU11	SECDU.SYS	TERDU.SYS
DUP11	SECDUP.SYS	TERDUP.SYS
DUV11	SECDUV.SYS	TERDUV.SYS
QNA	SECQNA.SYS	TERQNA.SYS
UNA	SECUNA.SYS	TERUNA.SYS

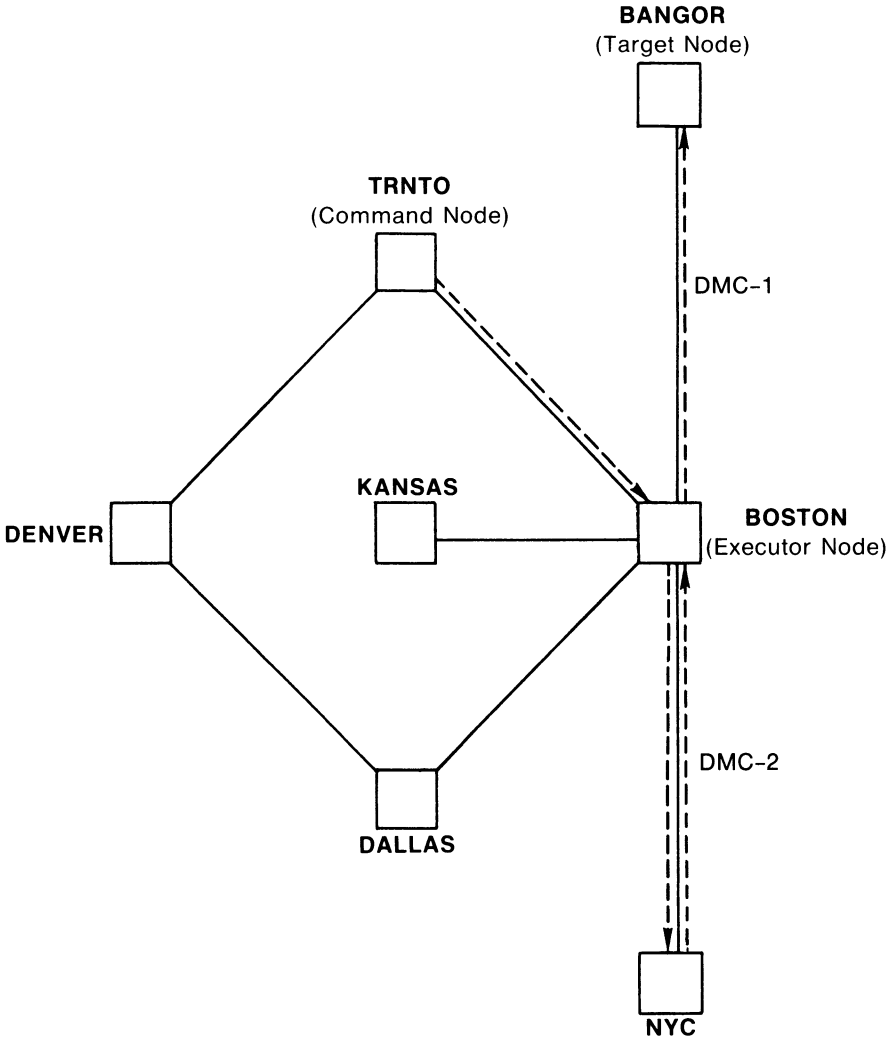
4.1.2.5 Management File Identification

The management file specifies a data file containing additional management information necessary for downline loading to a target node. You can supply a management file by specifying the MANAGEMENT FILE parameter with a LOAD NODE or LOAD VIA command. You can also establish the management file value in the node database using the SET NODE command.

For example:

```
NCP>SET NODE BANGOR MANAGEMENT FILE MANAGE.DAT
```

Figure 4–6 Operator-Initiated Downline Load over DDCMP Circuit (LOAD Command)



NCP Commands:

```
NCP>SET EXECUTOR NODE BOSTON
NCP>SET LINE DMC-1 STATE ON
NCP>SET CIRCUIT DMC-1 STATE ON
NCP>LOAD NODE BANGOR FROM NYC::MOM$LIBRARY:RSX11S.SYS
```

ZK-1868-84

4.1.2.6 Software Type

Along with identifying load files, you can specify the file types to be used for the initial load. For example, if the target node is already running a secondary loader program, you may only want to load the tertiary loader and operating system downline. To do this, you use the SOFTWARE TYPE parameter with the LOAD command. For example, to load a tertiary loader file, which in turn loads the operating system image, enter the following command:

```
NCP>LOAD NODE BANGOR SOFTWARE TYPE OPERATING SYSTEM
```

Use the SET NODE command to specify default software type information for the target node entry in the volatile database. If no software type information is specified in the volatile database, the default type is the secondary loader.

4.1.2.7 Load Assist Agent Identification

The load assist agent is the image that passes additional parameters to MOM to allow for downline loading to a node in a Local Area VAXcluster. To specify this image, use the node parameter LOAD ASSIST AGENT with a LOAD NODE or LOAD VIA command. You can also set the LOAD ASSIST AGENT value in the node database with the SET NODE command. The following command specifies a file containing a specific load assist agent:

```
NCP>LOAD NODE BANGOR LOAD ASSIST AGENT SYS$SHARE:NISCS_LAA.EXE
```

4.1.2.8 Load Assist Parameter Identification

Load assist agents pass parameters to MOM. To add to the set of parameters already contained in the load assist agent file, use the node parameter LOAD ASSIST PARAMETER. You can set this parameter value using the LOAD NODE, LOAD VIA, or SET NODE command. The following command passes an additional parameter to the load assist agent file:

```
NCP>SET NODE BANGOR LOAD ASSIST PARAMETER SYS$SYSDEVICE:[SYS9.]
```

4.1.2.9 CPU and Software Identification

The software identification is the default program name of the operating system to be loaded downline. Use the SOFTWARE IDENTIFICATION parameter to specify a *software-id* of up to 16 alphanumeric characters. For example, in the following command the CPU parameter specifies the default processor type to be loaded downline:

```
NCP>SET NODE BANGOR SOFTWARE IDENTIFICATION RSX_11S_V3.2
```

4.1.2.10 Service Device Identification

The service device is the controller on the target node end of a service circuit. The service device handles downline loading in a variety of ways, depending on the device used. In particular, this device influences the type of files suitable for downline loading. Default load file names are selected according to the service device for the target node.

The SERVICE DEVICE parameter identifies the default secondary and tertiary loaders for the load operation. This parameter is required for any downline load if the secondary and tertiary load files are not specified in the volatile database of the target node. SERVICE DEVICE is also required if the program load requests from the target node do not specify the secondary and tertiary load file names. Use the SET NODE command to specify the service device type in the volatile database. For example, the following command identifies the service device as a DMC device controller:

```
NCP>SET NODE BANGOR SERVICE DEVICE DMC
```

By using the SERVICE DEVICE parameter for the LOAD command, you can override the service device default information.

4.1.2.11 Service Circuit Identification

In terms of the executor, the service circuit is a circuit connecting the executor node with an adjacent target node. When you use the LOAD and TRIGGER commands, you must specify or default to a circuit over which the load operation is to take place. Use the VIA parameter to explicitly identify the circuit when entering these commands. If specifying an Ethernet circuit in the LOAD VIA command, you must include the PHYSICAL ADDRESS parameter.

If you do not specify a circuit, this information defaults to the circuit specified in the target node's volatile database. To set a service circuit in the volatile database, use the SET NODE command.

4.1.2.12 Service Passwords

When defining nodes for downline loading in the local volatile database, the system manager can specify a default service password. This password may be required to trigger the primary bootstrap mechanism on the target node. If you enter a LOAD or TRIGGER command without a service password, then this default parameter is used if the target node requires one. To set a service password in the volatile database, use the SET NODE command. This password must be a hexadecimal number in the following ranges:

- For DMC/DMR/DMP/DMV, the range is 0 to FFFFFFFF
- For UNA/QNA, the range is 0 to FFFFFFFFFFFFFFFF

For example, to load node BANGOR on circuit DMC–1, enter the following commands:

```
NCP>SET NODE BANGOR SERVICE PASSWORD FEFEFEFE
NCP>LOAD NODE BANGOR
```

4.1.2.13 Diagnostic File

After the target node is loaded downline, it can request diagnostics. Use the DIAGNOSTIC FILE parameter in the SET NODE command to identify in the volatile database the diagnostics file that the target node can read.

4.2 Dumping Memory Upline from an Unattended System

As a DECnet–VAX system manager, you can include certain SET NODE parameters in the volatile database that allow an adjacent unattended node to dump its memory into a file on your VMS operating system. This procedure is referred to as upline dumping. It is a valuable tool for crash analysis; that is, programmers can analyze the dump file and determine why the unattended system failed. When an unattended system that selected the appropriate support at system generation detects an impending system failure, that system requests an upline dump; for example, an RSX–11S operating system may request an upline memory dump to a VMS operating system.

For upline dump operations, the local VMS node is referred to as the executor and the adjacent unattended node as the slave.

4.2.1 Upline Dump Procedures

This section describes the procedures for an upline dump initiated by a slave node. DECnet uses the maintenance operation protocol (MOP) to perform an upline dump operation. MOP is a subset of the DIGITAL Data Communications Message Protocol (DDCMP) that sends messages used for circuit testing, triggering, downline loading, and upline dumping. Refer to the *Maintenance Operation Protocol Functional Specification* for a more complete discussion.

There are four steps involved in the upline dump process. The actual dump takes place when step 3 is repeated.

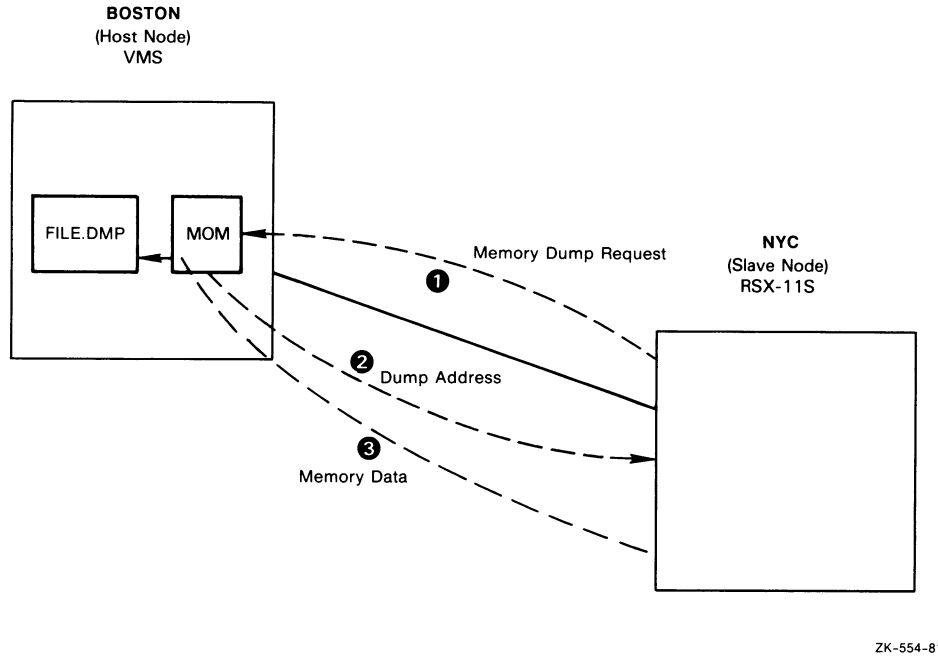
1. When a slave node senses a system failure, it sends a memory dump request to the VMS host node, or, on the Ethernet, to a dump assistance multicast address if an Ethernet host is not available. The request is a MOP request dump service message. This message may contain information about the slave's memory size (DUMP COUNT) and the upline dump device type at the slave.
2. If the message from the slave includes a DUMP COUNT value, the host node uses it. Otherwise, the host node checks the slave node's entry in its volatile database for the DUMP COUNT, the target address from which to start dumping (DUMP ADDRESS), and the file where the memory will be stored (DUMP FILE) for the slave. (If no entry exists for DUMP ADDRESS, the value defaults to 0.) The host node, which can now be considered the executor, sends a MOP request memory dump message to the slave with the starting address and buffer size values.
3. Using the values it receives from the executor, the slave returns the requested block of memory in a MOP memory dump data message. The executor receives the block of dump data, places it in the DUMP FILE, increments the DUMP ADDRESS by the number of locations sent, and sends another request memory dump message to the slave. This sequence is repeated until the amount of memory dumped matches the DUMP COUNT value. The executor then sends a MOP Dump Complete message to the target.
4. When the upline dump is complete, the executor node automatically attempts to downline load the slave system. It initiates the downline load by sending a TRIGGER message to the slave (see Section 4.1).

If the target node is on an Ethernet circuit, the target will attempt to perform an upline dump to the node that originally loaded it downline. If that node is not available, the target node proceeds as follows:

1. The target node sends a memory dump request to the Ethernet dump assistance multicast address AB-00-00-01-00-00 (described in Section 3.3.4.4). This message is a request for any node on the Ethernet to receive an upline memory dump.
2. The nodes on the Ethernet whose circuits are enabled to perform service functions check their own databases to determine if they can accept an upline dump. If so, they respond to the target node. The target chooses the node responding first to continue the dumping sequence. The target does not send a message to any other node. The loading sequence continues normally from there. Note, however, that you may have to look for event 0.3 in the event logs for all nodes on the Ethernet to determine which node received the dump. See the *VMS Network Control Program Manual* for a summary of all NCP events.

Figure 4–7 illustrates the upline dump procedure.

Figure 4–7 Upline Dumping of RSX–11S Memory



4.2.2 Upline Dump Requirements

Prior to attempting an upline dump operation, you must ensure that the nodes, lines, and circuits meet the following requirements:

- The slave node must be directly connected to the executor node by a physical line. The executor node provides the line- and circuit-level access.
- The slave node must be capable of requesting the upline dump when it detects a system failure. If the dumping program does not exist on the slave, upline dumping cannot occur.
- The circuit involved in the dump operation must be enabled to perform service functions. It must also be in the ON state. For example, the following command readies circuit DMC–0 for upline dumping node BANGOR in the network example:

```
NCP>SET CIRCUIT DMC-0 SERVICE ENABLED STATE ON
```

- If the slave does not supply the DUMP COUNT value, the executor must have this value in its volatile database.
- The executor must have a DUMP FILE entry in the volatile database. If the *file-id* specifies a remote node, the executor transfers the data using remote file access routines.

- Upline dumping cannot occur unless you define a service timer for the line. This timer sets the timeout constant for retransmission, enabling MOP to handle error recovery. For example, to set the retransmission frequency to 5000 milliseconds for line DMC–0, enter the following command:

```
NCP>SET LINE DMC-2 SERVICE TIMER 5000
```

4.3 Loading RSX–11S Tasks Downline

Downline task loading extends nonresident initial task load, checkpointing, and overlay support to a DECnet RSX–11S node. You can load an RSX–11S task downline by using the Satellite Loader (SLD) on the DECnet–11S node and the host loader (HLD) on the DECnet–VAX node. SLD uses the intertask communication facilities of RSX DECnet–11S to communicate with HLD. Figure 4–8 illustrates one instance of this relationship.

By entering RUN TLK at the operator's console of the satellite system, SLD requests HLD to load the task downline from a DECnet–VAX node on which the file is located. Any request from the satellite or host node could also initiate this operation by means of SLD and HLD.

4.3.1 Setting Up the Satellite System

You build the SLD task during the RSX–11S NETGEN procedure. (Refer to the *DECnet–RSX Network Generation and Installation Guide*.) To allow downline task loading, enter the appropriate commands to the RSX–11S system image. Use VMR to install and fix SLD into the RSX–11S system, as follows:

```
>VMR

ENTER FILENAME:RSX11S

VMR>INS SLD

VMR>FIX LDR...
```

This sequence of commands establishes SLD as the loading task (LDR...) for the executive.

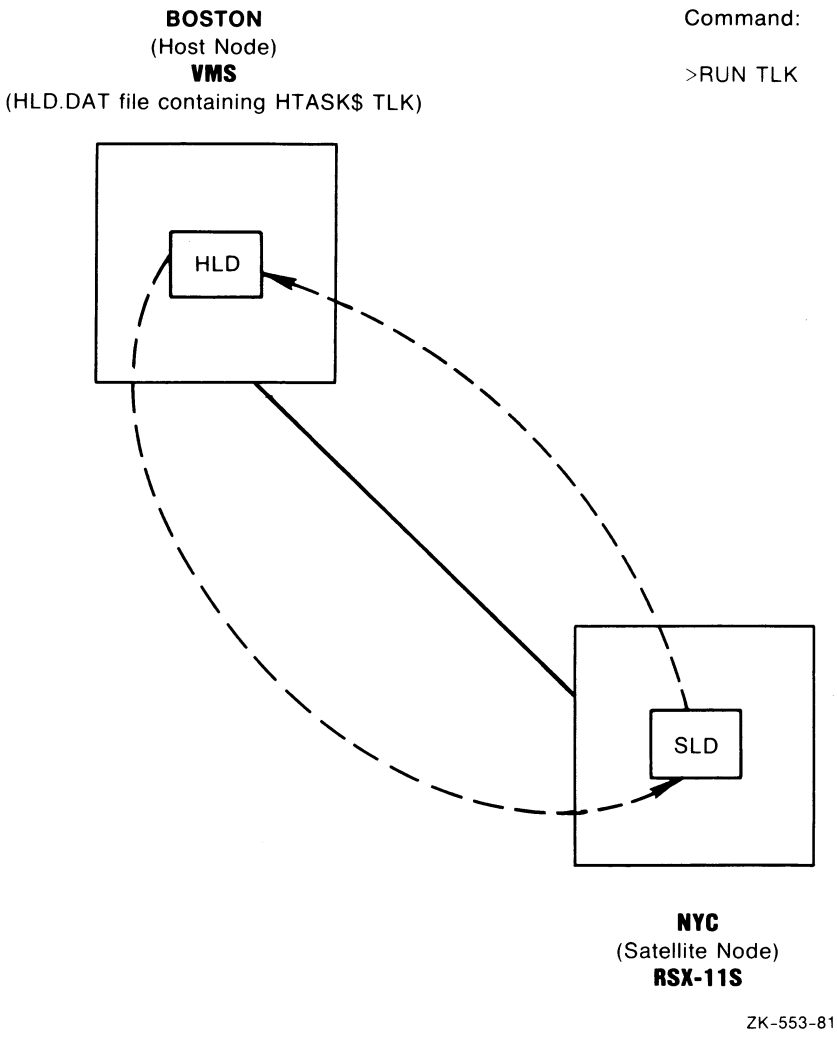
Note: The information in this section is specific to DECnet-RSX. For more information, refer to the related DECnet-RSX documentation.

If the RSX–11S system is to be loaded downline, any tasks to be downline loaded to or checkpointed from the RSX–11S system must be installed, but not fixed, using VMR. For example:

```
>VMR

ENTER FILENAME:RSX11S

VMR>INS TLK
```

In this example, entering RUN TLK on a terminal connected to the RSX–11S remote system initiates the downline task load of the file TLK.TSK.

Figure 4–8 Downline Task Loading



If the RSX–11S system will not be loaded downline, you must specify the node to which SLD will connect, using the VNP command SET EXECUTOR HOST (see the *DECnet–RSX Network Management Concepts and Procedures*). For example, you could use the following command, where 11 is the number of the node BOSTON on which HLD resides:

```
>VNP RSX11S
VNP>SET EXECUTOR HOST 11
```

4.3.2 Host Loader Mapping Table

The Host Loader has a mapping table that is a special user-defined file (HLD.DAT) that resides in the SYS$SYSTEM directory. The format of the mapping table is as follows:

```
HLDTB$
HTASK$  TLK,<TRNTO::SYS$DISK:[LOW]TLK>,UNM
HTASK$  TLK,<SYS$DISK:[LOW.EXT]TLK>,MAP
HNODE$  BANGOR
HTASK$  NCP,<SYS$DISK:[LOW]NCP11S>,LUN
HTASK$  ...LOA,<SYS$DISK:[TEST]LOA>
HNODE$  NYC
HTASK$  ...MCR,<B:[RSX11S.UNMAPPED]BASMCR>
.END
```

The following are keywords for this table.

HLDTB$ Defines the file as the HLD mapping table.

HTASK$ Defines a task entry. The arguments for HTASK$ are

taskname, <filespec> ,[opt-arg]

taskname	Is the installed task name used to run the task on the RSX–11S system.
filespec	Is the task file specification on the host node. You must use angle brackets (<>) to enclose the file specification.
opt-arg	Are optional arguments—MAP, UNM, LUN.

HNODE$ Defines the exclusive target node upon which the HTASK$ can execute.

This table is almost identical in structure to a MACRO–11 source module used by DECnet–RSX to define its downline task loading tables. Note, however, that HLD.DAT is accessed directly as a text file and is neither assembled nor task built. The organization of the mapping table and special features is as follows:

- A task entry contains the name of the installed task, a file specification, and an optional control argument. When you use the file specification in the HTASK$ macro, you can omit the file type that defaults to TSK. A node entry contains only the node name.
- Any task entries that precede the first node entry are called general-purpose tasks. You can load a general-purpose task into any RSX–11S node in the network. Task entries that follow a node entry can be sent only to that particular node.
- The same task name can appear more than once in the general-purpose task list. This allows both mapped and unmapped RSX–11S systems to share installed task names. The control argument for a general-purpose task is either MAP or UNM. The default is MAP.
- Tasks to be loaded downline must be installed in the RSX–11S system, which initializes the task's logical unit number (LUN) assignments. LUN-fixing is an SLD feature that reinitializes the LUNs after the task has been loaded downline. This feature allows a single task to be loaded into multiple RSX–11S systems that may have different systemwide device assignments. SLD permits you to place a task in a general-purpose task

list. You can downline load either a general-purpose task or a task after a node entry.

- If you place a task in a general-purpose task list, you can add new nodes to the network and can downline load general-purpose tasks to those nodes without changing the mapping table. Nodes that are to receive only general-purpose tasks need not be mentioned in HLD.DAT. Note, however, that general-purpose tasks cannot be checkpointed.

4.3.3 HLD Operation and Error Reporting

When SLD attempts to connect to HLD, NETACP on the DECnet–VAX node uses the default inbound access control information specified for the HLD object by the system manager (see Section 3.13). You must make sure that the files associated with the tasks to be loaded or checkpointed are accessible from the resulting process created by this connection.

When the load operation completes, whether successfully or unsuccessfully, the log file SYS$LOGIN:NETSERVER.LOG (described in Section 2.6.3) contains information describing the operation, the node, and the task. This information may consist of an error returned from RMS or certain HLD-specific messages that indicate either errors in HLD.DAT or inconsistencies in the file to be loaded. Messages associated with these inconsistencies are listed in the following section.

4.3.3.1 HLD Error Messages

The following is a list of HLD error messages.

Format error in HLD.DAT

The format of the HLD mapping table is incorrect. For example, this error could occur if HNODE$ was expected but not found in the table. Re-create the table, using the appropriate format.

Syntax error in HLD.DAT

The syntax of an element in the HLD mapping table is incorrect. For example, the angle brackets needed to enclose the file specification are missing. Re-create the table.

Task name not found

The task to be loaded downline is not specified in the HLD mapping table. Re-create the table so that it contains this task name.

No header in task file

The file was built with the /-HD switch. Therefore, it is an invalid RSX–11S task image. Rebuild the task.

Mapped task not on 4K boundary

The file was not built with the /MM switch. This error is for mapped RSX–11S systems only. Rebuild the task.

Unmapped partition mismatch

The TKB address does not correspond with the starting address of the partition in the RSX–11S system. This error is for unmapped RSX–11S systems only. Rebuild the task with a PAR= statement that specifies the correct starting address.

File too big for partition

The initial load size of the file is larger than the partition size in the RSX–11S system. Either make the partition larger or rebuild the file to use a smaller partition size.

Partition too big for checkpoint space

The partition size in the RSX–11S system is larger than the checkpoint space inside the file. Typically, this indicates that the partition size in your PAR= statement is smaller than the actual size of the partition in the RSX–11S system. Although the load size of a task may be much smaller than its partition, the entire partition is transferred during checkpoint operations. Rebuild the task with the exact partition size from the RSX–11S system.

4.3.4 Checkpointing RSX–11S Tasks

Checkpointing allows the execution of a task to be interrupted when a higher priority task installed in the same partition becomes active. The software writes the interrupted task from RSX–11S memory to a checkpoint file on the host (Checkpoint Write) and then loads the higher priority task into the partition and activates it. When the priority task exits, the software restores the interrupted task into main memory (Checkpoint Read), where it continues executing.

Note that checkpointing implies that a job is already running in the partition. Checkpoint space must be allocated inside the task being loaded downline (through the /AL switch during RSX–11S task build).

4.3.5 Overlaying RSX–11S Tasks

Overlaying allows the execution of segments of a task in order to reduce the memory or address space requirements for that task to run on an RSX–11S system. SLD and HLD handle the reading of overlay segments by satellite systems.

4.4 Connection to Remote Console

DECnet–VAX allows you to set up a logical connection between your VMS node and the console interface on certain unattended nodes, in effect permitting your terminal to act as the console for the remote system. For example, your terminal can act as the console for the DIGITAL Ethernet Communications Server (DECSA) hardware and its resident software, such as the Router Server. The console carrier requester on the host connects to the console carrier server on the server.

You can set up the logical connection to the console using the remote console facility (RCF). Both your host node and the target node (that is, the server node) must be on the same Ethernet. You can use the RCF to force a crash if the server node becomes unresponsive. (To determine how to force a crash, see the appropriate documentation for the particular server product.) RCF also permits debugging under special circumstances.

To use the RCF to connect to a DECSA, you must be sure the console carrier server image and its loader file are present in the system directory on the host node. (The file name of the console carrier server image is PLUTOCC.SYS and that of the loader is PLUTOWL.SYS.) To invoke RCF, specify either the CONNECT NODE or CONNECT VIA command. The VMS operating system then uses the loader file to downline load the console carrier server image into the Ethernet Communications Server hardware unit.

Use the CONNECT NODE command if the name of the target node is known. If the target node's service password and service circuit are defined in the host node's volatile database, you can use these default values. If the Ethernet hardware address of the server node is not defined in the volatile database, you must specify the PHYSICAL ADDRESS parameter in the CONNECT NODE command. If you specify the Ethernet physical address of the target node, DECnet–VAX attempts to use it to load the image file. If you do not supply an Ethernet address, DECnet–VAX derives an Ethernet physical address from the target node number, first attempting to use this address, and then attempting to use the Ethernet hardware address.

To define default information in the volatile database for the target node, use the NCP command SET NODE to specify the SERVICE PASSWORD, SERVICE CIRCUIT, and HARDWARE ADDRESS parameters for the target node. You can override the target node parameter values currently defined in the volatile database by specifying new values in the CONNECT command.

For example, to connect your VMS terminal to the console interface on server node RTRDEV, whose Ethernet physical address on circuit UNA–0 is AA–00–04–00–38–00, enter the following command:

```
NCP>CONNECT NODE RTRDEV SERVICE PASSWORD FEFEFEFEFEFEFEFE -
_ VIA UNA-0 PHYSICAL ADDRESS AA-00-04-00-38-00
```

Use the CONNECT VIA command if the node name of the target node is not known. In this command, you must specify the service circuit over which the logical connection is to be made and the Ethernet physical address of the target node.

If you have not defined the hardware address of the server node in the volatile database and have not specified the Ethernet physical address of the node in the CONNECT command, DECnet–VAX displays an error message on your terminal, as follows:

```
Hardware address required
```

This message indicates that you must specify an Ethernet address for the target node in your CONNECT command, because no hardware address is available in the volatile database.

In addition to the messages DECnet–VAX NCP or MOM may issue during downline loading of the console carrier server code, other messages may be issued when you attempt to connect to a remote console. For example:

```
Console in use
```

The remote console has already been reserved for another purpose. Try to make the connection later.

```
Console connected (press CTRL/D when finished)
```

The RCF is now ready for use. CTRL/B is used to pass a *break* character to the remote console. CTRL/D terminates the console session and causes the NCP prompt to be displayed.

```
Target does not respond
```

The remote console is supposed to respond quickly to inputs but is not doing so, or no connection can be made.

Part III Network Configuration, Installation, and Testing

5 Configuration of a Network

This chapter explains how to set up your VMS operating system for use in a DECnet–VAX network and provides sample configuration examples for various types of network. The *Guide to DECnet–VAX Networking* provides a summary of basic instructions for bringing up a DECnet–VAX node in the network.

5.1 Prerequisites for Establishing a Network

Before configuring your DECnet–VAX node, you need to satisfy certain prerequisites for DECnet–VAX operation, such as setting up user accounts and directories, defining user privileges, and registering the key to enable your DECnet–VAX license. (Note that you need a DECnet–VAX license only if you are planning to run DECnet in a multinode environment.)

If you are configuring VAX PSI, you must install it and define the privileges required for VAX PSI operation. If you are configuring VAX PSI in multihost mode (instead of native mode), you must install the VAX PSI multihost software on the VMS connector node that will serve as an X.25 gateway, and the VAX PSI Access software on each host node that will use the connector node.

5.1.1 User Accounts and Directories

In addition to creating normal user accounts in the user authorization file (UAF), you should also create a default nonprivileged DECnet account that can be used for activating network objects on the local node. DECnet–VAX uses the access privileges of this account when access control information has not been explicitly supplied by the network user. Section 3.13 discusses access control and the use of default accounts. Refer to the *Guide to Setting Up a VMS System* for a description of how to create and use directories.

The following example illustrates the commands you use to establish a default nonprivileged DECnet account. If you use NETCONFIG.COM to configure your node and request a default nonprivileged DECnet account, the account is created for you automatically (see Section 5.2.1.2).

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>ADD NETNONPRIV/PASSWORD=NONPRIV/DEVICE=DISK$USER1: -
_ /DIRECTORY=[NETUSER]/UIC=[200,200]/PRIVILEGE=(TMPMBX,NETMBX) -
_ /FLAGS=(CAPTIVE)/NOBATCH/NOINTERACTIVE/LGICMD=NL:
UAF>EXIT
$ CREATE/DIRECTORY DISK$USER1:[NETUSER]/OWNER_UIC=[200,200]
```

Note that you must substitute your own device name in place of DISK$USER1 when you set up the default nonprivileged DECnet account.

Configuration of a Network

5.1 Prerequisites for Establishing a Network

If you are configuring VAX PSI, ensure that you have the necessary PSI accounts, and the required directories associated with these PSI accounts, which will be used for incoming calls to your local DTE. You should also specify account information to activate objects at the local DTE for use by VAX PSI.

5.1.2 Required Privileges

To perform any kind of network activity, a process must have the privileges required to access the network processes involved in the activity requested. As system manager, you define a user's privileges in the UAF. The privileges listed in Table 5–1 are at times required by the Network Management Listener (NML) and by those users of DECnet–VAX running NCP. The privileges listed in Table 5–2 are required at times by both network users and system managers for VAX PSI operations.

Table 5–1 Required DECnet–VAX Privileges

Privilege	Description
ACNT	Allows you to create subprocesses or detached processes in which accounting is disabled. You need ACNT to start the network.
BYPASS	Allows you to view passwords that would not otherwise be displayed by the NCP command SHOW or LIST.
CMKRNL	Allows a process to change its access mode to kernel, execute a specified routine, and then return to its original access mode. Specifically, you need CMKRNL to start the network.
DETACH	Allows you to create detached processes by executing the $CREPRC system service. You need DETACH to bring up the network.
NETMBX	Allows you to assign a channel to the NET device. You need this privilege to create a logical link or to perform any ACP control QIO functions. NETMBX is the minimum requirement for all accounts running network programs.
OPER	Allows you to perform certain operator functions such as modifying the configuration database. (Refer to the *Guide to Setting Up a VMS System* for a detailed explanation of the functions available under this privilege.) The NML needs this privilege to modify any network parameters in the volatile database.
TMPMBX	Allows you to create temporary mailboxes. If you have this privilege, you can use the $CREMBX and $ASSIGN system services to create a temporary mailbox and assign an I/O channel for task-to-task communication. Unlike a permanent mailbox, which must be explicitly deleted, a temporary mailbox is automatically deleted when no more channels are assigned to it. TMPMBX is required for the default accounts and to run both NML and NCP.
SYSNAM	Allows you to declare a name or object number in a user task (see Chapter 8 for information about user tasks).
SYSPRV	Allows you to access the permanent database.

Table 5–2 Required VAX PSI Privileges

Privilege	Description
SECURITY	Allows a system manager to access the VAX PSI security databases.
DIAGNOSE	Allows a system manager to use diagnostic functions. You can use it to run online diagnostic programs. You need DIAGNOSE to perform line loop tests.
NETMBX	Allows a network user to assign a channel to the NW device. This channel is required to set up virtual circuits or to perform any ACP control QIO functions. Either NETMBX privilege or the PSI$X25_USER right is required for processes running network programs.
OPER	Allows a system manager to use the NCP commands. The *VMS Network Control Program Manual* provides a detailed explanation of the NCP functions available under this privilege.
TMPMBX	Allows a network user to create temporary mailboxes, that is, to use the $ASSIGN system service to create a temporary mailbox and assign an I/O channel for DTE-to-DTE communication. Unlike a permanent mailbox, which must be deleted explicitly, a temporary mailbox is deleted automatically when no more channels are assigned to it.
SYSPRV	Allows a system manager to create and display objects.

Refer to Section 3.13 for a further discussion of network user privileges and their function in relation to overall network security.

5.2 Configuration Procedures

Before you install DECnet–VAX on your system, there are certain tasks that you, as system manager, must complete to prepare for a networking environment.

You must configure your DECnet–VAX permanent database. You can use the interactive procedure NETCONFIG.COM provided by Phase IV DECnet–VAX to do this. NETCONFIG.COM prompts you for all the information needed to configure the permanent database and to set up an optional default nonprivileged DECnet account on your system. If you choose not to use NETCONFIG.COM, you must use NCP commands to build the permanent database. Alternatively, you can use NCP commands to tailor the permanent database created by NETCONFIG.COM to your own needs. Also, you can use the NCP command COPY KNOWN NODES to build or update the remote node entries in your node database. Section 5.4.5 discusses special considerations that apply to configuration of the permanent database for a VAXcluster node.

You may have to perform additional configuration tasks depending upon your specific network requirements. If you plan to run DECnet–VAX over a CI, you must install the DECnet driver CNDRIVER. If you will be using some of your terminal lines as DECnet–VAX lines, you must install the asynchronous DDCMP driver NODRIVER and set up the static or dynamic asynchronous lines. Section 5.2.2 describes these tasks in detail.

If you are planning to run DECnet–VAX on a VAXcluster, you must take special care when setting the SYSGEN parameters SCSNODE and SCSSYSTEMID. You must set SCSNODE to match the executor node name. You must set SCSSYSTEMID to match the executor address. When setting SCSSYSTEMID, use the algorithm for converting a node address to its decimal equivalent, as explained in Section 3.7.2. Section 5.4 provides additional information about setting up SYSGEN parameters..

If VAX PSI is to be run, the system manager is responsible for configuring VAX PSI for the local DTEs. This involves supplying information about various VAX PSI components, such as circuits, lines, modules, and objects. The information is contained in the DECnet–VAX configuration database for the local node (if both DECnet–VAX and PSI are configured) and in the PSI configuration database for the local DTEs. You use NCP commands to supply information to the configuration database.

5.2.1 Using NETCONFIG.COM

The NETCONFIG.COM command procedure performs the following steps:

1. Prompts you for the name and address of your node and asks whether or not you want a default nonprivileged DECnet account and whether you want to operate as a router or an end node.
2. Determines which DECnet devices you have on your system.
3. Creates and displays the NCP and DCL commands required to configure your DECnet–VAX node.
4. Executes the commands displayed, if they are accepted, setting the appropriate parameters in the permanent configuration database at your node. This procedure establishes all databases (executor, line, circuit, object, logging) except for the remote node database, and purges any existing information from the permanent database.

Use of this optional procedure is recommended when you are bringing up a new system as a DECnet–VAX node or when you want to completely revise the configuration database for a system that is already running DECnet–VAX.

Specifically, if you are bringing up a new system, follow these steps:

1. Execute NETCONFIG.COM, replying NO to the question about starting the network.
2. Use NCP commands to modify or add parameters after the initial database is configured. To make changes in or add DECnet user and proxy accounts, use the DCL command AUTHORIZE.
3. Set up the remote node database using NCP DEFINE NODE commands.
4. Execute STARTNET.COM to bring up your DECnet–VAX node (see Section 6.2).

If you are running DECnet–VAX on a node and want to completely revise all permanent configuration databases except the remote node database, you should execute NETCONFIG.COM. To make any further alterations in the databases and DECnet user and proxy accounts, use NCP commands and the DCL command AUTHORIZE.

If you are running DECnet–VAX and want to preserve the existing permanent database, do not use NETCONFIG.COM. Use NCP commands to make any desired changes in the database.

5.2.1.1 Executing NETCONFIG.COM

You must have system privilege (SYSPRV) to execute NETCONFIG.COM. To invoke the command procedure, enter the following command:

```
$ @SYS$MANAGER:NETCONFIG.COM
```

The only information you must supply to the procedure is the node name and node address of your system. The node name is a string of up to six alphanumeric characters, containing at least one alphabetic character. The node address is a numeric value in the format:

area-number.node-number

where:

area-number Designates the area in which the node is grouped (in the range 1 to 63).

node-number Designates the node's unique address within the area (in the range 1 to 1023).

If the network is not divided into two or more areas, you do not have to provide the area number; the system supplies the default area number 1.

The procedure also asks whether you want a default nonprivileged DECnet account to be established for you. If you indicate yes (or take the default YES by pressing the RETURN key), the procedure displays the DCL AUTHORIZE command that creates a default nonprivileged DECnet account with a null password and a UIC of [376,376]. (Note that you can change this UIC value or other account parameters by using the Authorize Utility after the node is configured.)

NETCONFIG.COM then asks if you want the executor node to function as a router. If you type NO (or take the default NO by pressing the RETURN key) in response to this question, the executor node is set up as a nonrouting node.

NETCONFIG.COM automatically determines which DECnet devices exist on your system for use in building the line and circuit permanent databases. NETCONFIG.COM then uses the information you supply and the information it obtains about the system to create all the commands necessary to configure your system as a DECnet–VAX node, and displays these commands for your approval (see Section 5.2.1.2.) The commands define the permanent database parameters for the executor; all lines, circuits, and objects; and all logging monitor events. The commands do not define the database for any remote node.

Inspect the displayed commands. NETCONFIG.COM asks if you want to configure using these commands. If you answer yes, the procedure establishes the permanent configuration database and default nonprivileged DECnet account (if you requested it). If you answer no, the procedure returns a message indicating that no changes have been made.

Answer yes to the final question, which asks if you want the network started automatically. You should have already registered the DECnet–VAX key.

If you have purchased a DECnet–VAX license but have not yet registered the appropriate DECnet–VAX key for that license, do so now. You can then start up the network manually by entering the following command:

```
$ @SYS$MANAGER:STARTNET
```

After the permanent database is established, you have the option of using NCP commands to alter the parameters to correspond more closely to your configuration requirements.

If you use NETCONFIG.COM to establish the configuration database for a system that will be using asynchronous lines (for example, a MicroVAX system with a terminal line), NETCONFIG.COM does not configure the asynchronous circuit and line parameters automatically. Instead, NETCONFIG.COM displays a message indicating that no circuits or lines have been configured. You must set up the asynchronous lines separately (see Section 5.2.2.2). Also, NETCONFIG.COM does not set up CI circuits.

To ensure that the installation is successful, you can use the User Environment Test Package (UETP) to test DECnet. The test procedure is described in the *Guide to Setting Up a VMS System*.

5.2.1.2 NETCONFIG.COM Example

The following example shows the interactive dialog that is displayed when you execute NETCONFIG.COM to configure node CHCAGO.

```
            DECnet-VAX network configuration procedure

This procedure will help you define the parameters needed to get
DECnet running on this machine.  You will be shown the changes before
they are actually executed, in case you wish to perform them manually.

What do you want your DECnet node name to be?          :  CHCAGO
What do you want your DECnet address to be?            :  2.37
Do you want to operate as a router?   [NO (nonrouting)]:  RET
Do you want a default DECnet account?            [YES]:  RET

            Here are the commands necessary to set up your system.

---------------------------------

$ RUN SYS$SYSTEM:NCP
     PURGE EXECUTOR ALL
     PURGE KNOWN LINES ALL
     PURGE KNOWN CIRCUITS ALL
     PURGE KNOWN LOGGING ALL
     PURGE KNOWN OBJECTS ALL
     PURGE MODULE CONFIGURATOR KNOWN CIRCUITS ALL
$ DEFINE/USER SYS$OUTPUT NL:
$ DEFINE/USER SYS$ERROR NL:
$ RUN SYS$SYSTEM:NCP
    PURGE NODE 2.37 ALL
    PURGE NODE CHCAGO ALL
$ RUN SYS$SYSTEM:NCP
    DEFINE EXECUTOR ADDRESS 2.37 STATE ON
    DEFINE EXECUTOR NAME CHCAGO
    DEFINE EXECUTOR MAXIMUM ADDRESS 1023
    DEFINE EXECUTOR ROUTING TYPE NONROUTING IV
    DEFINE EXECUTOR NONPRIVILEGED USER DECNET
$ DEFINE/USER SYSUAF SYS$SYSTEM:SYSUAF.DAT
$ RUN SYS$SYSTEM:AUTHORIZE
    ADD DECNET /OWNER="DECNET DEFAULT" -
        /PASSWORD=DECNET -
        /UIC=[376,376] /ACCOUNT=DECNET -
```

```
        /DEVICE=SYS$SYSDEVICE: /DIRECTORY=[DECNET] -
        /PRIVILEGE=(TMPMBX,NETMBX)
        /FLAGS=(CAPTIVE)/NOBATCH/NOINTERACTIVE /LGICMD=NL: -
        /NOBATCH /NOINTERACTIVE
$ CREATE/DIRECTORY SYS$SYSDEVICE:[DECNET] /OWNER=[376,376]
$ RUN SYS$SYSTEM:NCP
    DEFINE LINE    CI-0  STATE ON
    DEFINE LINE    UNA-0 STATE ON
    DEFINE CIRCUIT UNA-0 STATE ON COST 3
    DEFINE LINE    DMC-0 STATE ON
    DEFINE CIRCUIT DMC-0 STATE ON COST 5
    DEFINE LINE    DMC-1 STATE ON
    DEFINE CIRCUIT DMC-1 STATE ON COST 5
    DEFINE LOGGING MONITOR STATE ON
    DEFINE LOGGING MONITOR EVENTS 0.0-9
    DEFINE LOGGING MONITOR EVENTS 2.0-1
    DEFINE LOGGING MONITOR EVENTS 4.2-13,15-16,18-19
    DEFINE LOGGING MONITOR EVENTS 5.0-18
    DEFINE LOGGING MONITOR EVENTS 128.0-4

-----------------------------------

Do you want to go ahead and do it?                [YES]: Y
The changes have been made.

If you have not already registered the DECnet-VAX key, then do so now.

After the key has been registered, you should invoke the procedure
SYS$MANAGER:STARTNET.COM to startup DECnet-VAX with these changes.

(If the key is already registered) Do you want DECnet started? [YES]
```

5.2.2 Tailoring the Configuration Database

If you do not choose to use NETCONFIG.COM to build the network configuration in the permanent database, you may instead configure the network using individual NCP commands. Examples of various configuration procedures are given in Section 5.3. You can also use NCP to add or delete entries in an existing permanent database.

Following are two examples of changes made to the network configuration that require corresponding modification of the permanent database:

- Running DECnet over the CI. The driver CNDRIVER must be loaded on the system and all CI lines and circuits must be defined in the configuration database.
- Running DECnet over terminal lines. The terminal driver, NODRIVER, must be loaded on the system, terminal lines must be converted to DDCMP lines, and all DDCMP lines and circuits must be defined in the configuration database.

The procedures for handling these changes are described in detail in the following sections.

5.2.2.1 Running DECnet over the CI

To use the CI750, CI780, or CIBCI as a DECnet device on your VMS operating system, you must first install CNDRIVER, the DECnet driver associated with the CI. To load CNDRIVER, add the following commands to the SYSTARTUP_V5.COM command procedure in the SYS$MANAGER directory:

```
$ RUN SYS$SYSTEM:SYSGEN
CONNECT CNA0/NOADAPTER
```

You are now ready to run DECnet over the CI. The following example illustrates how to use NCP commands to define the CI line and one or more CI circuits in the permanent database.

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE LINE CI-0 STATE ON
NCP>DEFINE CIRCUIT CI-0.0 TRIBUTARY 0 STATE ON
NCP>DEFINE CIRCUIT CI-0.1 TRIBUTARY 1 STATE ON
    .
    .
    .
NCP> EXIT
$
```

5.2.2.2 Running DECnet over Terminal Lines

To use lines connected to terminal ports as DECnet communications lines, you must load the asynchronous DDCMP driver NODRIVER, set up the terminal lines to be converted to asynchronous DDCMP lines, and specify the appropriate lines and circuits in the NCP configuration database. The steps in converting terminal lines to asynchronous lines depend on the type of line you want to set up:

- A static asynchronous DDCMP line: a line permanently configured as a DECnet line
- A dynamic asynchronous DDCMP line: a line that is switched from terminal to DECnet use for the duration of a dialup call

Procedures for installing and shutting down each of these types of lines are described in Section 5.2.2.3 and Section 5.2.2.4. The complete DECnet–VAX installation procedure, including establishment of asynchronous connections, appears in the *Guide to DECnet–VAX Networking*.

Because dialup lines are more prone to noise problems than dedicated synchronous lines, you should set the executor buffer size and segment buffer size to a value of 192 for any end node that is connected to its router by a dialup line. Use of a relatively small buffer size reduces the effect of buffer retransmission on overall throughput.

5.2.2.3 Installing Static Asynchronous Lines

You perform the following steps when setting up and shutting down static asynchronous lines on your system.

Setting Up Static Asynchronous DDCMP Lines

The following steps are necessary to set up lines connected to terminal ports on your system for use as static asynchronous DECnet lines. Note that the system manager at the remote node must also perform these steps.

1 Load the asynchronous DDCMP driver NODRIVER by adding the following commands to the SYSTARTUP_V5.COM command procedure in the SYS$MANAGER directory or by specifying the commands after the system is booted.

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT NOA0/NOADAPTER
```

2 Choose the terminal lines on your system that you will use as DECnet lines. To convert a line to a static asynchronous DDCMP line, add the DCL command SET TERMINAL/PROTOCOL=DDCMP device-name to SYSTARTUP_V5.COM in your SYS$MANAGER directory for each terminal line that will run DECnet. Insert the SET TERMINAL command before the command @SYS$MANAGER:STARTNET in SYSTARTUP_V5.COM.

For example, to convert the terminal lines connected to ports TTA0 and TXB7 on your system into DECnet lines with no modem control, add the following commands to SYSTARTUP_V5.COM:

```
$ SET TERMINAL/PERMANENT/PROTOCOL=DDCMP/NOTYPE_AHEAD  TTA0:
$ SET TERMINAL/PERMANENT/PROTOCOL=DDCMP/NOTYPE_AHEAD  TXB7:
```

To convert the line connected to terminal port TXA1 (which can be used as a dialup line) to a DECnet line with modem control, add the following command to SYSTARTUP_V5.COM:

```
$ SET TERMINAL/PERMANENT/MODEM/NOHANGUP/NOAUTOBAUD -
_$ /NOTYPE_AHEAD/PROTOCOL=DDCMP TXA1:
```

Note that while the terminal line is in use as a DECnet communications line, you can change the line speed by resetting the speed and line using NCP.

3 Use NCP commands to define all terminal lines and circuits in the configuration permanent database, as shown in the following example:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE LINE TT-0-0 STATE ON RECEIVE BUFFERS 4 -
_ LINE SPEED 2400
NCP>DEFINE CIRCUIT TT-0-0 STATE ON
NCP>DEFINE LINE TX-1-7 STATE ON RECEIVE BUFFERS 4 -
_ LINE SPEED 2400
NCP>DEFINE CIRCUIT TX-1-7 STATE ON

NCP>DEFINE LINE TX-1-1 STATE ON RECEIVE BUFFERS 4 -
_ LINE SPEED 2400
NCP>DEFINE CIRCUIT TX-1-1 STATE ON
NCP>EXIT
$
```

The lines are then turned on to the network.

Reasons for Failure of Static Asynchronous Connections

If static asynchronous DECnet lines are started but are left in the ON-STARTING state, check the following:

- The line speeds at both ends of the connection must be set to the same value.
- If you are using a dialup line, the modem characteristic must be set on the terminal before the line is used for asynchronous DDCMP.
- If the network is divided into areas, the two nodes being connected must be in the same area or must be area routers.
- Asynchronous DECnet requires the parity on the asynchronous line to be set to NONE and the terminal line to be set up to use 8-bit characters. If you are using a non-VMS system, you must check that the terminal line is set to the correct parity.

If your terminal line cannot be set up as a static asynchronous DDCMP line, check whether the following condition exists:

- If data is stored in a type-ahead buffer associated with your terminal line, the line comes up as a terminal line even if a startup command procedure attempts to set it up as a DDCMP line. This generally occurs when the remote node is running and its asynchronous DDCMP line is on. The DDCMP start messages being transmitted are stored in the type-ahead buffer for your line. Before you can start up your terminal line in DDCMP mode, you must terminate the process that has started and that owns your terminal line.

To verify that the asynchronous line is connected properly, check the following:

- For local connections, verify that the cable is a null modem cable.
- For modem connections, verify that the cable is a straight-through cable and that if the modem is put in analog loopback, the circuit comes up with the local node as the adjacent node.
- For both types of connections, verify that the port is operational by resetting the port to terminal-type characteristics and plugging in a terminal and logging in.

Shutting Down Static Asynchronous DDCMP Lines

To shut down a DECnet line and return it to a terminal line, enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE TT-0-0 STATE OFF
NCP>CLEAR LINE TT-0-0 ALL
NCP>SET CIRCUIT TT-0-0 STATE OFF
NCP>CLEAR CIRCUIT TT-0-0 ALL
```

To switch a line for which modem control was not enabled back to a terminal line, enter the following command:

```
$ SET TERMINAL/PROTOCOL=NONE TTA0:
```

To switch a line for which modem control was enabled back to a terminal line, enter the following command:

```
$ SET TERMINAL/PERMANENT/MODEM/AUTOBAUD/TYPEAHEAD -
_$ /PROTOCOL=NONE TXA1:
```

5.2.2.4 Installing Dynamic Asynchronous Lines

To make a temporary connection to another node over an asynchronous connection (for example, a telephone line), the terminal lines at each node may be switched to dynamic asynchronous DDCMP lines for the duration of the connection. The procedure for establishing a dynamic connection, reasons why the connection might fail, and the actions that shut down the lines are described next. Dynamic switching is described in detail in Chapter 2.

Setting Up Dynamic Asynchronous DDCMP Lines

You perform the following steps to set up a dynamic connection, and to switch lines connected to terminal ports to dynamic asynchronous DECnet lines. This procedure illustrates commands used if a local VMS operating system installed on a MicroVAX (WRKVAX) is establishing a dynamic connection with a remote VMS operating system (BIGVAX). The remote system is the node that performs the switch.

1 The system manager at each node must load the asynchronous DDCMP driver NODRIVER by adding the following commands to the SYSTARTUP_V5.COM command procedure in the SYS$MANAGER directory or by specifying the commands after the system is booted.

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT NOA0/NOADAPTER
```

2 The system manager at each node must install the shareable image DYNSWITCH, as follows:

```
$ INSTALL:=$SYS$SYSTEM:INSTALL
$ INSTALL/COMMAND
INSTALL> CREATE SYS$LIBRARY:DYNSWITCH/SHARE -
_ /PROTECT/HEADER/OPEN
INSTALL> EXIT
```

Note that DYNSWITCH is a DECnet–VAX image only. If the image DYNSWITCH is not installed on the remote system, dynamic switching of lines is implicitly disabled.

3 The system manager at the remote node, BIGVAX, must enable the use of virtual terminals with these commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT VTA0/NOADAPTER/DRIVER=TTDRIVER
```

(For further information on using virtual terminals, refer to the *Guide to Maintaining a VMS System*.)

The system manager on the remote system must also enable the disconnect option for the terminal port to be used by specifying the following command for the terminal:

```
$ SET TERMINAL/PERMANENT/MODEM/DISCONNECT TTB1:
```

4 For security, the user at the local node WRKVAX must define in the node database the transmit password to be sent to remote node BIGVAX. For example:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE BIGVAX TRANSMIT PASSWORD password
```

The system manager at remote node BIGVAX must define node WRKVAX in the node database with the appropriate receive password and INBOUND type (router or end node). For example:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE WRKVAX INBOUND ENDNODE RECEIVE PASSWORD password
```

5 The user at the local system, WRKVAX, must perform the following steps:

a. Log in to the local system.

b. Enter the following command to establish a system process running a terminal emulator (TTA1 identifies the terminal port on the local system through which the dynamic connection will be made):

```
$ SET HOST/DTE TTA1:
```

You can optionally include the /DIAL qualifier in the SET HOST command to cause automatic dialing of the modem to the remote node.

c. Dial in to the remote system.

d. Log in to your account on the remote node BIGVAX.

e. If DYNSWITCH is installed on your local system, you can enter the following command to initiate automatic dynamic switching at both ends of the connection:

```
$ SET TERMINAL/PROTOCOL=DDCMP/SWITCH=DECNET
```

You receive the following message when DECnet is started at the local node:

```
%REM-S-END, control returned to node_local-node-name::
$
```

Use the line for network operations when you see the local system prompt ($).

f. Alternatively, you can switch the line manually. Specify the /MANUAL qualifier in the SET TERMINAL command:

```
$ SET TERMINAL/PROTOCOL=DDCMP/SWITCH=DECNET/MANUAL
```

You receive the following message from the remote system:

```
%SET-I-SWINPRG, The line you are currently logged
                  over is becoming a DECnet line
```

Exit the terminal emulator by pressing the Backslash (\) key and the CTRL key simultaneously.

```
$ SET TERMINAL/PROTOCOL=DDCMP TTA1:
```

Enter the following NCP commands to turn on your line and circuit:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE TT-0-1 RECEIVE BUFFERS 4 STATE ON
$ RUN SYS$SYSTEM:NCP
NCP>EXIT
```

DECnet is now started at the local node.

Note that the SET TERMINAL command is a VMS DCL command. If you are on a non-VMS node (for example, if you are running MS-DOS on a personal computer), you should specify the equivalent function for your system.

Reasons for Failure of Dynamic Asynchronous Connections

If dynamic switching is being performed and the asynchronous DECnet connection is not made, first check that the following conditions exist:

- DECnet must be started on both nodes.
- The asynchronous DDCMP class driver (NODRIVER) must be loaded by means of SYS$SYSTEM:SYSGEN at each node.
- The dynamic switch image (DYNSWITCH) must be installed by means of SYS$SYSTEM:INSTALL at each VMS node.
- Virtual terminals must be enabled both on the remote node and, in particular, for the terminal at which you are logged in. The terminal line at the remote node must have the attribute DISCONNECT set.
- After you enter a SET TERMINAL command with the /MANUAL qualifier, you must specify NCP commands to turn on the DECnet line within approximately four minutes or the line returns to terminal mode.

If the dynamic asynchronous lines are started but are left in the ON-STARTING state, check the following:

- If the network is divided into areas, the two nodes being connected must be in the same area or must be area routers.
- The routing initialization passwords on each node must be set correctly (see Section 3.13.1).
- The INBOUND parameter for the local node entry must be set correctly in the node database at the remote node (see Section 3.13.3).
- Asynchronous DECnet requires the parity on the asynchronous line to be set to NONE and the terminal line to be set up to use 8-bit characters. If you are using a non-VMS system, you must check that the terminal line is set to the correct parity.

Shutting Down Dynamic Asynchronous Lines

You have two options for ending a dynamic connection:

- Break the telephone connection.

- Enter one of the following NCP commands (either command causes both line and circuit entries to be cleared in the database):

```
NCP>SET LINE TT-0-0 STATE OFF

NCP>SET CIRCUIT TT-0-0 STATE OFF
```

The results of these commands vary depending on the side of the connection from which they are entered. If the command is entered at the local (originating) node, the port is immediately switched to the terminal driver. On the other side (the remote node), the line remains in the ON-STARTING state for approximately four minutes and then is switched to the terminal port. If the line or circuit is stopped by the remote node, the line and circuit on both sides of the connection immediately return to terminal mode.

Note that, if you specify the /NOHANGUP qualifier in the SET TERMINAL command with which you initiate dynamic switching, the modem carrier signal is not dropped when you shut down the DECnet line or circuit. The carrier signal is broken when you hang up the telephone.

If you specify the SET TERMINAL command with the /MANUAL qualifier to switch the terminal line manually, you can abort the switch by pressing CTRL/C or CTRL/Y.

5.3 Network Configuration Examples

This section discusses how you can use NCP commands to build your network configuration in the permanent database. The following subsections show examples of the NCP commands you can use to obtain the following network configurations:

- Synchronous DDCMP point-to-point configuration
- DDCMP multipoint configuration
- Static asynchronous DDCMP point-to-point configuration
- Dynamic asynchronous DDCMP point-to-point configuration
- Ethernet configuration
- X.25 DLM configuration
- X.25 native mode configuration
- X.25 multihost mode configuration
- X.25 multinetwork configuration

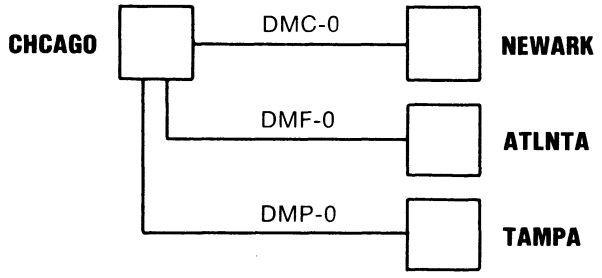
You should assume that these configuration examples are single-area networks using the default area 1. For an example of the NCP commands used to configure a multiple-area network, see Section A.3. Figure 5–1 through Figure 5–9 correspond to the examples shown in each of the respective sections.

Combine the appropriate NCP commands in a command file that reflects your network configuration; then edit and run this procedure as many times as necessary to properly build the permanent database corresponding to your configuration and needs. After you configure the permanent database, invoke SYS$MANAGER:STARTNET.COM to load these parameters into the volatile database, and to bring up the network.

5.3.1 Synchronous DDCMP Point-to-Point Network Example

The example in this section shows how to build a database for a network configuration of four nodes connected by a DMC11, DMP11, or DMF32 line and circuit. The NCP commands in this example configure the DDCMP point-to-point network. Note that node NEWARK is a nonrouting RSX-11S system to which node CHCAGO will perform a downline load.

Figure 5–1 A Synchronous DDCMP Point-to-Point Network Configuration



ZK-1852-84

```
!
!  Define executor-specific parameters for local node CHCAGO.
!  The TYPE parameter for the executor node defaults to
!  ROUTING IV.
!

 DEFINE EXECUTOR            ADDRESS 1  -
                            BUFFER  SIZE 576  -
                            MAXIMUM  HOPS 6  -
                            MAXIMUM  VISITS 12  -
                            STATE ON
!
!  Define common node parameters for the local node.  Be sure
!  to add the NETNONPRIV user to your system authorization
!  file by using the Authorize Utility.
!

 DEFINE  EXECUTOR           NAME CHCAGO  -
                            NONPRIVILEGED  -
                                  USER NETNONPRIV  -
                                  PASSWORD NONPRIV
```

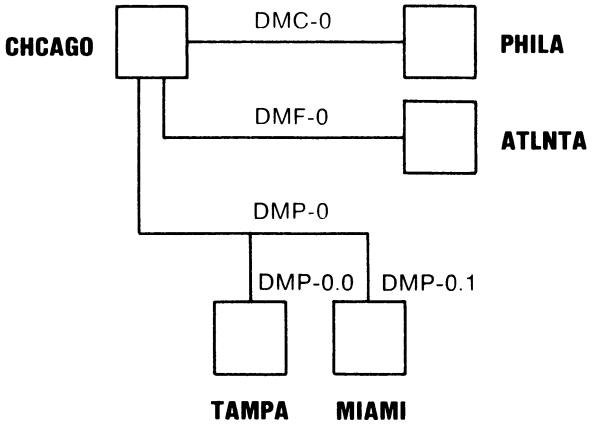
```
!
!  Define parameters for remote node NEWARK  (a nonrouting
!  RSX-11S system).  CHCAGO will be the load host for NEWARK.
!
 DEFINE  NODE  2            NAME NEWARK  -
                            HOST NODE CHCAGO  -
                            LOAD FILE NOD11S.SYS  -
                            NONPRIVILEGED  -
                                   USER  NETNONPRIV  -
                                   PASSWORD  NONPRIV  -
                            SERVICE CIRCUIT DMC-0  -
                            SERVICE PASSWORD FE  -
                            SECONDARY LOADER SECDMC.SYS  -
                            TERTIARY LOADER TERDMC.SYS

!
!  Define the remaining nodes.  Note that no default outbound
!  access control information is specified.  This assumes that
!  the default access control information will be supplied by
!  each remote node when it receives an inbound connect request.
!
 DEFINE NODE 3              NAME ATLNTA
 DEFINE NODE 4              NAME TAMPA

!
!  Define parameters for line/circuit DMC-0 to node NEWARK.
!
!  Because this node will be loaded downline, the service
!  parameters must be set up.
!
 DEFINE  LINE  DMC-0        PROTOCOL DDCMP POINT  -
                            SERVICE TIMER 4000  -
                            STATE ON

 DEFINE CIRCUIT DMC-0       SERVICE ENABLED  -
                            STATE ON

!
!  Define parameters for line/circuit DMF-0 to node ATLNTA.
!
!  (Give this line more receive buffers because it has a faster
!  connection.)
!
 DEFINE  LINE  DMF-0        PROTOCOL DDCMP POINT  -
                            RECEIVE BUFFERS 8  -
                            STATE  ON

 DEFINE CIRCUIT DMF-0       STATE ON

!
!  Define parameters for line/circuit DMP-0 to node TAMPA.
!
 DEFINE  LINE   DMP-0       PROTOCOL DDCMP POINT -
                            STATE ON

 DEFINE CIRCUIT DMP-0       STATE ON

!
!  The object database does not need to be defined because it defaults
!  to the standard list of objects known to the VMS operating system.
!
```

```
!
!  Define transmitter-related logging parameters.
!
 DEFINE LOGGING MONITOR KNOWN EVENTS
!
!  Define receiver-related logging parameters.
!
 DEFINE LOGGING MONITOR STATE ON
```

5.3.2 DDCMP Multipoint Network Example

The example in this section shows how to build a database for a network configuration of five nodes connected by a combination of DMC, DMF, and DMP lines and circuits. The NCP commands in this example configure the DDCMP multipoint network.

Figure 5–2 A DDCMP Multipoint Network Configuration




```
!
!  Define executor-specific parameters for local node CHCAGO.
!  The TYPE parameter for the executor node defaults to
!  ROUTING IV since a full function license is installed.
!
 DEFINE EXECUTOR          ADDRESS 1  -
                          BUFFER  SIZE 576  -
                          MAXIMUM  HOPS 6  -
                          MAXIMUM  VISITS 12  -
                          STATE ON
```

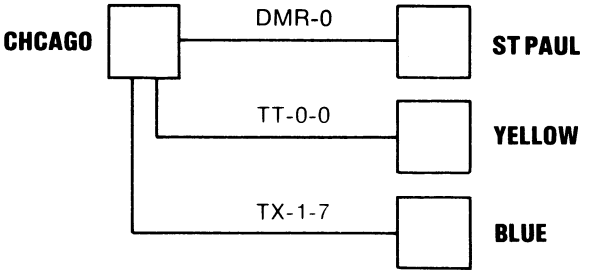
```
!
!  Define common node parameters for the local node.  Be sure
!  to add the NETNONPRIV user to your system authorization
!  file by using the Authorize Utility.
!
 DEFINE  EXECUTOR          NAME CHCAGO  -
                           NONPRIVILEGED  -
                                USER NETNONPRIV  -
                                PASSWORD NONPRIV

!
!  Define the remaining nodes.  Note that no default outbound
!  access control information is specified.  This assumes that
!  the default access control information will be supplied by
!  each remote node when it receives an inbound connect request.
!
 DEFINE NODE 2             NAME PHILA
 DEFINE NODE 3             NAME ATLNTA
 DEFINE NODE 4             NAME TAMPA
 DEFINE NODE 5             NAME MIAMI

!
!  Define parameters for line/circuit DMC-0 to node PHILA.
!
 DEFINE  LINE  DMC-0       PROTOCOL DDCMP POINT  -
                           STATE ON  -
 DEFINE CIRCUIT DMC-0      STATE ON

!
!  Define parameters for line/circuit DMF-0 to node ATLNTA.
!
 DEFINE  LINE  DMF-0       PROTOCOL DDCMP POINT  -
                           STATE  ON
 DEFINE CIRCUIT DMF-0      STATE ON

!
!  Define parameters for line DMP-0 and circuits to nodes  TAMPA
!  and MIAMI.
!
!            TAMPA is connected as tributary 3, DMP-0.0
!            MIAMI is connected as tributary 4, DMP-0.1
!
!  The DMP line runs at 56,000 bits per second.  The proper
!  setting for the retransmit timer is
!
!                                  20,000 * buffer_size
!            retransmit timer =    --------------------
!                                           bps
!
!  Thus, with a buffer size of 576, the retransmit timer should
!  be 210 milliseconds.
!
!  The dead timer is set to 30 seconds to avoid excessive delays
!  when polling dead tributaries.  The timer is set when a node
!  goes down.
!
 DEFINE  LINE  DMP-0       PROTOCOL DDCMP CONTROL  -
                           DEAD TIMER 30000  -
                           RECEIVE BUFFERS 6  -
                           RETRANSMIT TIMER 210 -
                           STATE  ON
```

```
DEFINE CIRCUIT DMP-0.0  COST 4  -
                        STATE ON  -
                        TRIBUTARY 3

DEFINE CIRCUIT DMP-0.1  COST 4  -
                        STATE ON  -
                        TRIBUTARY 4

!
!  The object database does not need  to  be defined because it defaults
!  to the standard list of objects known to the VMS operating system.
!

!
!  Define transmitter-related logging parameters.
!

DEFINE LOGGING MONITOR KNOWN EVENTS

!
!  Define receiver-related logging parameters.
!

DEFINE LOGGING MONITOR STATE ON
```

5.3.3 Static Asynchronous DDCMP Network Example

The example in this section shows how to build a database for a network configuration of four nodes connected by a DMR11 line and two terminal lines converted to static asynchronous DECnet lines. The NCP commands in this example configure the DDCMP point-to-point network that includes static asynchronous lines. To establish the static asynchronous connections, nodes YELLOW and BLUE must also specify in their configuration databases the circuits and lines connecting them to node CHCAGO. Before entering these commands, you should set up the terminal line with the appropriate characteristics (see Section 5.2.2.3).

Figure 5–3 A Static Asynchronous DDCMP Network Configuration



ZK-1854-84

```
!
!  Define executor-specific parameters for local node
!  CHCAGO.  The TYPE parameter for the executor node defaults to
!  ROUTING IV.
!
 DEFINE EXECUTOR            ADDRESS 1  -
                            BUFFER  SIZE 576  -
                            MAXIMUM  HOPS 6  -
                            MAXIMUM  VISITS 12  -
                            STATE ON
!
!  Define common node parameters for the local node.  Be
!  sure to add the NETNONPRIV user to your system
!  authorization file by using the Authorize Utility.
!
 DEFINE  EXECUTOR           NAME CHCAGO  -
                            NONPRIVILEGED  -
                                   USER NETNONPRIV  -
                                   PASSWORD NONPRIV

!
!  Define the remaining nodes.  Note that no default
!  outbound access control information is specified.
!  This assumes that the default access control
!  information will be supplied by each remote node
!  when it receives an inbound connect request.
!
 DEFINE NODE 2              NAME STPAUL
 DEFINE NODE 3              NAME YELLOW
 DEFINE NODE 4              NAME BLUE

!
!  Define parameters for line/circuit DMR-0 to node
!  STPAUL.
!
 DEFINE  LINE  DMR-0        PROTOCOL DDCMP POINT  -
                            STATE ON

 DEFINE CIRCUIT DMR-0       STATE ON

!
!  Define parameters for line/circuit TT-0-0 to node
!  YELLOW.
!
 DEFINE  LINE  TT-0-0       RECEIVE BUFFERS 4  -
                            STATE  ON -
                            LINE SPEED 9600

 DEFINE CIRCUIT TT-0-0      STATE ON

!
!  Define parameters for line/circuit TX-1-7 to node
!  BLUE.
!
 DEFINE  LINE  TX-1-7       RECEIVER BUFFERS 4  -
                            STATE ON -
                            LINE SPEED 1200

 DEFINE CIRCUIT TX-1-7      STATE ON
```

```
!
!  The  object database  does  not  need  to  be defined
!  because it defaults to the standard list of objects
!  known to the VMS operating system.
!
!
!  Define transmitter-related logging parameters.
!
 DEFINE LOGGING MONITOR KNOWN EVENTS

!
!  Define receiver-related logging parameters.
!
 DEFINE LOGGING MONITOR STATE ON
```

5.3.4 Dynamic Asynchronous DDCMP Network Example

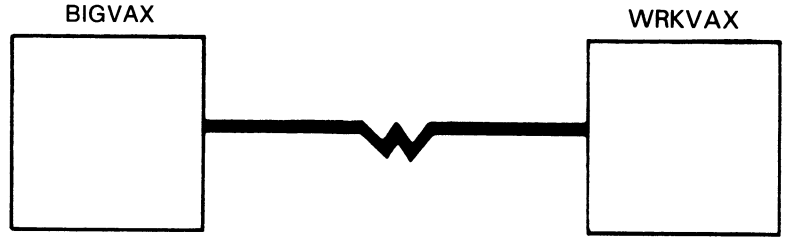
The examples in this section show how to configure two nodes connected by a terminal line converted to a dynamic asynchronous DECnet line.

The first example shows the NCP commands that configure the dynamic asynchronous DDCMP connection from node BIGVAX to node WRKVAX; node BIGVAX is assumed to be a router.

The second example shows the NCP commands that configure the dynamic asynchronous DDCMP connection from node WRKVAX to node BIGVAX; node WRKVAX is assumed to be an end node, and the dynamic line is assumed to be a slow (1200 baud) modem line.

Before entering these commands, refer to the procedure for installing dynamic asynchronous lines in Section 5.2.2.4.

Figure 5–4 A Dynamic Asynchronous DDCMP Network Configuration



Node BIGVAX Database

```
!
!  Define executor-specific parameters for local node
!  BIGVAX.
!

 DEFINE EXECUTOR            ADDRESS 1  -
                            BUFFER  SIZE 576  -
                            MAXIMUM  HOPS 6  -
                            MAXIMUM  VISITS 12  -
                            STATE ON
!
!  Define common node parameters for the local node.  Be
!  sure to add the NETNONPRIV user to your system
!  authorization file by using the Authorize Utility.
!

 DEFINE  EXECUTOR            NAME BIGVAX  -
                             NONPRIVILEGED  -
                                    USER NETNONPRIV  -
                                    PASSWORD NONPRIV

!
!  Define the remote node.  You must use the INBOUND
!  parameter to check whether dialup node WRKVAX will
!  operate as an end node or as a router.  As an added
!  security feature for a node using a dynamic asynchronous
!  communications line, you must also specify a receive
!  password for node WRKVAX.  This will be compared with
!  the transmit password supplied by WRKVAX when it
!  issues the connect request.
!

 DEFINE NODE 2               NAME WRKVAX -
                             INBOUND ENDNODE -
                             RECEIVE PASSWORD 10101010

!
!  You do not need to define parameters for a
!  line/circuit to node WRKVAX.  These parameters are
!  provided automatically by the system when the dynamic
!  connection is initiated.
!
!
!  The  object database  does  not  need  to  be defined
!  because it defaults to the standard list of objects
!  known to the VMS operating system.
!
!
!  Define transmitter-related logging parameters.
!

 DEFINE LOGGING MONITOR KNOWN EVENTS

!
!  Define receiver-related logging parameters.
!

 DEFINE LOGGING MONITOR STATE ON
```

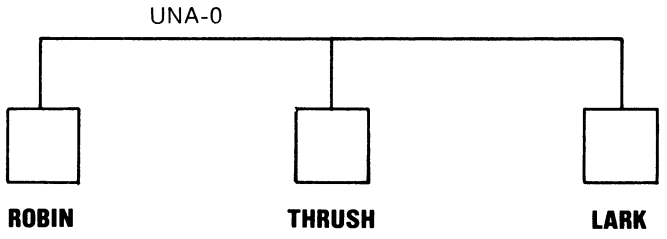
Node WRKVAX Database

```
!
!  Define executor-specific parameters for local node
!  WRKVAX.
!

 DEFINE EXECUTOR            ADDRESS 2  -
                            BUFFER  SIZE 192  -
                            SEGMENT BUFFER SIZE 192 -
                            STATE ON -
                            TYPE NONROUTING
!
!  Define common node parameters for the local node.
!  Be sure to add the NETNONPRIV user to your system
!  authorization file by using the Authorize Utility.
!

 DEFINE  EXECUTOR            NAME WRKVAX  -
                             NONPRIVILEGED  -
                                    USER NETNONPRIV  -
                                    PASSWORD NONPRIV

!
!  Define the remote node.  You must specify a transmit
!  password which matches the receive password in the
!  remote node database on node BIGVAX.
!

 DEFINE NODE 1               NAME BIGVAX -
                             TRANSMIT PASSWORD 10101010

!
!  You do not need to define parameters for a
!  line/circuit to node BIGVAX.  These parameters are
!  provided automatically by the system when the dynamic
!  connection is initiated.
!
!
!  The  object database  does  not  need  to  be defined
!  because it defaults to the standard list of objects
!  known to the VMS operating system.
!
```

5.3.5 Ethernet Network Example

The example in this section shows how to build a database for a network configuration of three nodes connected by an Ethernet UNA line and circuit. The NCP commands in this example configure the database for node ROBIN. Repeat the procedure to configure the databases for nodes THRUSH and LARK.

Figure 5–5 An Ethernet Network Configuration



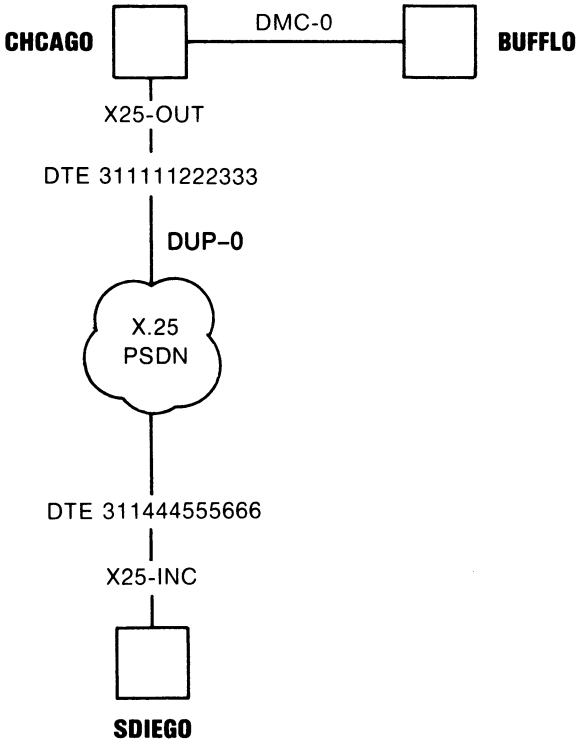

```
!
!  Define executor-specific parameters  for  local  node  ROBIN.
!  Note that the TYPE parameter for the executor node defaults
!  to a node type that corresponds  to  the  type  of  network
!  license (router or end node) you have installed.
!
 DEFINE EXECUTOR          ADDRESS 20  -
                          BUFFER  SIZE 576  -
                          STATE ON

!
!  Define common node parameters for the local node.  Be sure
!  to add the NETNONPRIV user to your system authorization
!  file by using the Authorize Utility.
!
 DEFINE  EXECUTOR         NAME ROBIN  -
                          NONPRIVILEGED  -
                               USER NETNONPRIV  -
                               PASSWORD NONPRIV

!
!  Define the remaining nodes.  Note that no default outbound
!  access control information is specified.  This assumes that
!  the default access control information will be supplied by
!  each remote node when it receives an inbound connect request.
!
 DEFINE NODE 21           NAME THRUSH
 DEFINE NODE 22           NAME LARK
!
!  Define parameters for line/circuit UNA-0.
!
 DEFINE  LINE  UNA-0      STATE ON
 DEFINE CIRCUIT UNA-0     STATE ON

!
!  The object database  does  not need  to  be  defined because  it defaults
!  to the standard list of objects known to the VMS operating system.
!
```

```
!
!  Define transmitter-related logging parameters.
!
 DEFINE LOGGING MONITOR KNOWN EVENTS

!
!  Define receiver-related logging parameters.
!
 DEFINE LOGGING MONITOR STATE ON
```

5.3.6 X.25 Data Link Mapping Example

The examples in this section show how to build a database for a network configuration of three nodes connected by a DMC11 line and circuit and an X.25 packet switching data network (PSDN). The first example shows the NCP commands that configure the database for node CHCAGO. The second example shows the NCP commands that configure the database for node SDIEGO.

Figure 5–6 An X.25 Data Link Mapping Network Configuration



ZK-1856-84

Node CHCAGO Database

```
!
!  Set up the X.25 protocol module.
!
 DEFINE MODULE X25-PROTOCOL  -
                        NETWORK TELENET PROFILE TELENET
!
!  Define the line used to communicate with the X.25 network.
!
 DEFINE LINE DUP-0       NETWORK TELENET STATE ON
!
!  Define the local DTE for node CHCAGO.
!
 DEFINE MODULE X25-PROTOCOL  -
                        DTE 311111222333  -
                        NETWORK TELENET -
                        CHANNELS 2018-1546  -
                        LINE DUP-0 -
                        STATE ON
!
!  Define executor specific parameters for local node CHCAGO.
!  The TYPE parameter for the executor node defaults to
!  ROUTING IV.
!
 DEFINE EXECUTOR         ADDRESS 1  -
                        BUFFER  SIZE 576  -
                        MAXIMUM  HOPS 6  -
                        MAXIMUM  VISITS 12  -
                        STATE ON

!
!  Define common node parameters for the local node.  Be sure
!  to add the NETNONPRIV user to your system authorization
!  file by using the Authorize Utility.
!
 DEFINE  EXECUTOR        NAME CHCAGO  -
                        NONPRIVILEGED  -
                               USER NETNONPRIV  -
                               PASSWORD NONPRIV

!
!  Define the remaining nodes.  Note that no default outbound
!  access control information is specified.  This assumes that
!  the default access control information will be supplied by
!  each remote node when it receives an inbound connect request.
!
 DEFINE NODE 2           NAME BUFFLO
 DEFINE NODE 3           NAME SDIEGO

!
!  Define parameters for line/circuit DMC-0 to node BUFFLO.
!
 DEFINE  LINE  DMC-0     PROTOCOL DDCMP POINT  -
                        STATE ON  -
 DEFINE CIRCUIT DMC-0    STATE ON
```

```
!
!  Define parameters for the outgoing DLM circuit X25-OUT to node
!  SDIEGO (node SDIEGO is addressed as DTE 311444555666 to the
!  X.25 network; subaddress 1 is defined on node SDIEGO to be a
!  DECnet DLM subaddress).
!

 DEFINE CIRCUIT X25-OUT -
                                NETWORK TELENET -
                                NUMBER 3114445556661 -
                                OWNER EXECUTOR  -
                                USAGE OUTGOING  -
                                STATE ON

!
!  The object database does not need  to  be  defined because it defaults
!  to the standard list of objects known to the VMS operating system.
!

!
!  Define transmitter-related logging parameter.
!

 DEFINE LOGGING MONITOR KNOWN EVENTS

!
!  Define receiver-related logging parameters.
!

 DEFINE LOGGING MONITOR STATE ON
```

Node SDIEGO Database

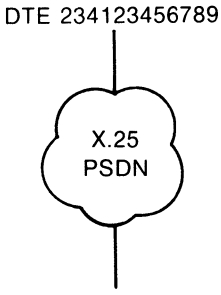
```
!
!  Set up the X.25 protocol module.
!

 DEFINE MODULE X25-PROTOCOL  -
                        NETWORK TELENET PROFILE TELENET
!
!  Define the line used to communicate with the X.25 network.
!
 DEFINE LINE DUP-0       NETWORK TELENET  STATE ON
!
!  Define the local DTE for node SDIEGO.
!

 DEFINE MODULE X25-PROTOCOL  -
                         DTE 311444555666  -
                         NETWORK TELENET -
                         CHANNELS 2490-2452  -
                         LINE DUP-0 -
                         STATE ON
!
!  Define executor-specific parameters  for  local node SDIEGO.
!  Note that the X.25 subaddress range is given as 1 to 5 so
!  that this node can accept all X.25 calls with a subaddress
!  from 1 to 5.  Because CHCAGO is sending X.25 calls and does
!  not intend to receive any, you need not specify the
!  subaddress parameter for this DTE.  The system manager must
!  coordinate the subaddress values used to designate DECnet
!  data link calls among the DECnet nodes.
!
```

```
DEFINE EXECUTOR         ADDRESS 3  -
                        BUFFER  SIZE 576  -
                        MAXIMUM  HOPS 6  -
                        MAXIMUM  VISITS 12  -
                        SUBADDRESSES 1-5  -
                        STATE ON

!
!  Define common node parameters for the local node.  Be sure
!  to add the NETNONPRIV user to your system authorization
!  file by using the Authorize Utility.
!
DEFINE  EXECUTOR        NAME SDIEGO  -
                        NONPRIVILEGED  -
                               USER NETNONPRIV  -
                               PASSWORD NONPRIV

!
!  Define the remaining nodes.  Note that no default outbound
!  access control information is specified.  This assumes that
!  the default access control information will be supplied by
!  each remote node when it receives an inbound connect request.
!
DEFINE NODE 1            NAME CHCAGO
DEFINE NODE 2            NAME BUFFLO

!
!  Define parameters for the incoming DLM circuit X25-INC (to
!  node CHCAGO).
!
DEFINE CIRCUIT X25-INC   NETWORK TELENET OWNER EXECUTOR  -
                         USAGE INCOMING  -
                         STATE ON

!
!  The object database does not need to be defined because it defaults
!  to the standard list of objects known to the VMS operating system.
!

!
!  Define transmitter-related logging parameters.
!
DEFINE LOGGING MONITOR KNOWN EVENTS

!
!  Define receiver-related logging parameters.
!
DEFINE LOGGING MONITOR STATE ON
```

5.3.7 X.25 Native Mode Network Example

The example in this section shows how to build a database for a network configuration associating a local DTE with a packet switching data network by a DUP line. The NCP commands in this example configure the permanent database for the X.25 native-mode network. These commands build server and object modules to handle incoming calls.

Figure 5–7 An X.25 Native-Mode Network Configuration




```
!
!  Set up the X.25 protocol module.
!
 DEFINE MODULE X25-PROTOCOL -
                        NETWORK SONNET PROFILE SONNET
!
!  Define the line used to communicate with the X.25 network.
!
 DEFINE LINE DUP-0       NETWORK SONNET STATE ON
!
!  Define the local DTE.

 DEFINE MODULE X25-PROTOCOL  -
                        DTE 234123456789 -
                        NETWORK SONNET -
                        CHANNELS 2018-1546 -
                        LINE DUP-0 -
                        STATE ON
!
!  Define the destinations.
!

 DEFINE MODULE X25-SERVER -
                        DESTINATION JOE -
                        SUBADDRESS 1-10 -
                        OBJECT OBJONE -
                        PRIORITY 1

 DEFINE MODULE X25-SERVER -
                        DESTINATION JIM -
                        SUBADDRESS 11-15 -
                        OBJECT OBJTWO -
                        PRIORITY 2

 DEFINE MODULE X25-SERVER -
                        DESTINATION DEFDEST -
                        OBJECT DEFOBJ -
                        PRIORITY 0

!
!  Define the X.29 call handler.
!

 DEFINE MODULE X29-SERVER STATE ON

!
!  Define the objects used for incoming calls.
!
```

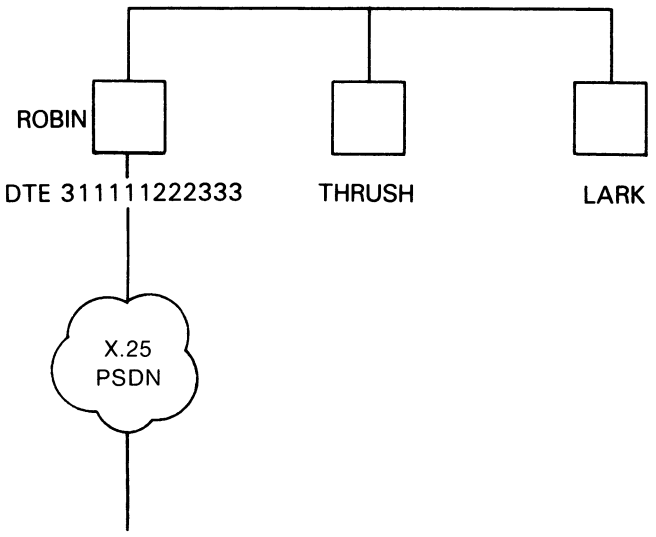
```
DEFINE OBJECT OBJONE -
                        FILE OBJSTUP.COM -
                        USER PSIUSER -
                        PASSWORD PSIUSER
DEFINE OBJECT OBJTWO -
                        FILE OBJECTTWO.COM -
                        USER JIM -
                        PASSWORD JIM
DEFINE OBJECT DEFOBJ -
                        FILE LSTCHNCE.COM -
                        USER NET -
                        PASSWORD NET
```

5.3.8 X.25 Multihost Mode Network Example

The examples in the following subsections illustrate the NCP commands used to configure three VMS nodes connected by an Ethernet UNA line and circuit, with one node set up as an X.25 multihost connector node to the PSDN called SONNET and the other two nodes set up as host nodes to use the connector node.

In these examples, node ROBIN, with VAX PSI multihost software installed, is configured as the connector node, and nodes THRUSH and LARK, each with VAX PSI Access software installed, are configured as host nodes that can communicate with the X.25 network through node ROBIN.

Figure 5–8 An X.25 Multihost Mode Network Configuration



To configure this network, you must complete the following tasks:

1. Build the Ethernet network.
2. Build the appropriate databases on ROBIN to configure this node as the X.25 connector node.
3. Configure THRUSH and LARK as host nodes capable of accessing the X.25 network through the connector node ROBIN.

5.3.8.1 Building the Ethernet Network

The following NCP commands configure the database for node ROBIN. Repeat this procedure to configure the databases for nodes THRUSH and LARK.

```
!
!  Define  executor-specific   parameters   for  local  node
!  ROBIN.   The TYPE parameter for the executor node defaults
!  to  ROUTING IV.
!

 DEFINE EXECUTOR          ADDRESS 20  -
                          BUFFER  SIZE 576  -
                          STATE ON

!
!  Define common node parameters for the local node.  Be sure
!  to add the NETNONPRIV user to your system authorization
!  file by using the Authorize Utility.
!

 DEFINE  EXECUTOR         NAME ROBIN  -
                          NONPRIVILEGED  -
                               USER NETNONPRIV  -
                               PASSWORD NONPRIV

!
!  Define the remaining nodes.
!

 DEFINE NODE 21           NAME THRUSH
 DEFINE NODE 22           NAME LARK

!
!  Define parameters for line/circuit UNA-0.
!

 DEFINE  LINE  UNA-0      STATE ON

 DEFINE CIRCUIT UNA-0     STATE ON

!
!  The  object database  does  not  need to be defined because it
!  defaults to the standard list of objects known to VMS.
!

!
!  Define transmitter-related logging parameters.
!

 DEFINE LOGGING MONITOR KNOWN EVENTS

!
!  Define receiver-related logging parameters.
!

 DEFINE LOGGING MONITOR STATE ON
```

5.3.8.2 Configuring the X.25 Connector Node

The following NCP commands build the X25-PROTOCOL and X25-SERVER databases for a multihost PSI node. Node ROBIN is configured as the VAX PSI multihost node connected to the PSDN named SONNET.

```
!
!  Set up the X.25 protocol module.
!
 DEFINE MODULE X25-PROTOCOL NETWORK PUBLIC PROFILE SONNET
!
!  Define the line used to communicate with the X.25 network.
!
 DEFINE LINE DUP-0 NETWORK PUBLIC STATE ON
!
!  Define the local DTE.
!

 DEFINE MODULE X25-PROTOCOL -
                      DTE 311111222333 -
                      NETWORK PUBLIC -
                      CHANNELS 2018-1546 -
                      LINE DUP-0 -
                      STATE ON
!
!  Define host destinations for incoming calls.
!

 DEFINE MODULE X25-SERVER -
                      DESTINATION THRUSH -
                      SUBADDRESSES 1-10 -
                      OBJECT 36 -
                      NODE THRUSH

 DEFINE MODULE X25-SERVER -
                      DESTINATION LARK -
                      SUBADDRESSES 11-20 -
                      OBJECT 36 -
                      NODE LARK
```

5.3.8.3 Configuring the Host Nodes

The following NCP commands build the X25-ACCESS and X25-SERVER databases on nodes THRUSH and LARK to allow both host nodes to access SONNET through connector node ROBIN.

Node THRUSH Database

```
!
!  Set up the X.25 access module.
!

 DEFINE MODULE X25-ACCESS -
                      NETWORK PUBLIC -
                      NODE ROBIN
!
!  Define the destination on THRUSH.
!

 DEFINE MODULE X25-SERVER -
                      DESTINATION JOE -
                      SUBADDRESS 1-10 -
                      OBJECT OBJONE
```

```
!
!  Define the X.29 call handler.
!

 DEFINE MODULE X29-SERVER STATE ON

!
!  Define the destination objects for incoming calls.
!

 DEFINE OBJECT OBJONE -
                      FILE OBJSTUP.COM -
                      USER PSIUSER -
                      PASSWORD PSIUSER
```

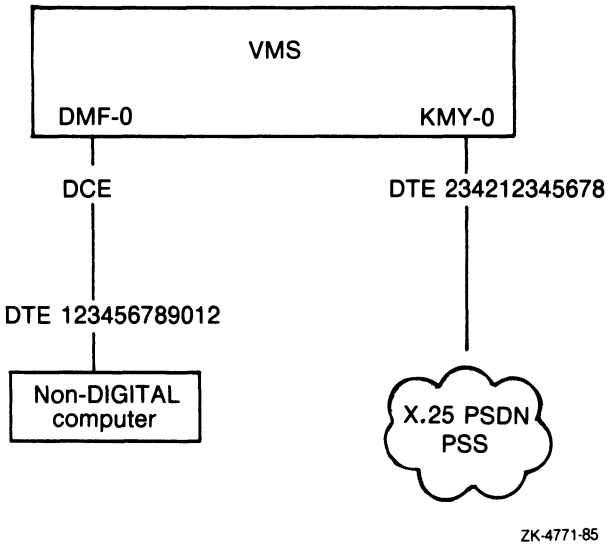
Node LARK Database

```
!
!  Set up the X.25 access module.
!

 DEFINE MODULE X25-ACCESS -
                      NETWORK PUBLIC -
                      NODE ROBIN

!
!  Define the destination on LARK.
!

 DEFINE MODULE X25-SERVER -
                      DESTINATION JOE -
                      SUBADDRESS 11-20 -
                      OBJECT OBJTWO

!
!  Define the X.29 call handler.
!

 DEFINE MODULE X29-SERVER STATE ON

!
!  Define the destination object for incoming calls.
!

 DEFINE OBJECT OBJTWO -
                      FILE OBJTWO.COM -
                      USER JIM -
                      PASSWORD JIM
```

5.3.9 X.25 Multinetwork Example

The example in this section shows how to build a database for a network configuration where a VMS operating system is connected to a PSDN and also to a non-DIGITAL machine that behaves like an X.25 DTE. The non-DIGITAL connection uses the profile ISO8208. VAX PSI is running in native mode. The NCP commands in this example configure the permanent database for the multinetwork configuration.

Figure 5–9 A Multinetwork Configuration




```
!
!  Set up the networks.  The PSDN connection will be called network PUBLIC
!  and the other network will be called  NONDEC (using the profile ISO8208).
!

 DEFINE MODULE X25-PROTOCOL -
                      NETWORK PUBLIC -
                      PROFILE SONNET

 DEFINE MODULE X25-PROTOCOL -
                      NETWORK NONDEC -
                      PROFILE ISO8208
!
!  Define the line used to connect to SONNET.
!

 DEFINE LINE KMY-0 NETWORK PUBLIC STATE ON

!
!  Define the line used to connect to the non-DIGITAL machine.
!

 DEFINE LINE DMF-0 -
                      NETWORK NONDEC -
                      INTERFACE DCE -
                      STATE ON
!
! Define the DTE addresses.
!

 DEFINE MODULE X25-PROTOCOL -
                      DTE 234212345678 -
                      NETWORK PUBLIC -
                      LINE KMY-0 -
                      CHANNELS 1056-1024
```

```
DEFINE MODULE X25-PROTOCOL -
                      DTE 123456789012 -
                      NETWORK NONDEC -
                      LINE DMF-0 -
                      CHANNELS 100-1 -
                      INTERFACE DCE
!
!  Define the destinations.  The  destination, JOE, receives calls
!  from PUBLIC only.
!
 DEFINE MODULE X25-SERVER -
                      DESTINATION JOE -
                      SUBADDRESS 1-10 -
                      NETWORK PUBLIC -
                      OBJECT OBJONE -
                      PRIORITY 1
!
!  The  destination, JIM, receives calls from either network.
!
 DEFINE MODULE X25-SERVER -
                      DESTINATION JIM -
                      SUBADDRESS 11-16 -
                      OBJECT OBJTWO -
                      PRIORITY 2
!
!  The destination, DEFDEST, provides a general destination for calls
!  that do not match the other destinations.
!
 DEFINE MODULE X25-SERVER -
                      DESTINATION DEFDEST
                      OBJECT DEFOBJ -
                      PRIORITY 0
!
!  Define the X.29 call handler.
!
 DEFINE MODULE X29-SERVER STATE ON

!
!  Define the objects used for incoming calls.
!
 DEFINE OBJECT OBJONE -
                      FILE OBJSTUP.COM -
                      USER PSIUSER -
                      PASSWORD PSIUSER
 DEFINE OBJECT OBJTWO -
                      FILE OBJECTTWO.COM -
                      USER JIM -
                      PASSWORD JIM
 DEFINE OBJECT DEFOBJ -
                      FILE LSTCHNCE.COM -
                      USER NET -
                      PASSWORD NET
```

5.4 System Configuration Guidelines

Proper network operation, particularly in a routing environment, requires that you properly configure the system software running on each node in the network. Memory and processor time are two principal resources that you need to define.

5.4.1 Normal Memory Requirements

Most of the memory required by the network software is allocated from the VMS nonpaged dynamic memory pool. You configure this pool by setting the SYSGEN parameters NPAGEDYN, IRPCOUNT, LRPCOUNT, and LRPSIZE.

These SYSGEN parameters are set by the AUTOGEN Facility when the operating system is first booted and do not normally require modification. If, however, you find that you need to modify the SYSGEN parameters to tune the system properly, you should edit the file SYS$SYSTEM:MODPARAMS.DAT as described in the *Guide to Maintaining a VMS System.*

5.4.1.1 NPAGEDYN Parameter

To increase the nonpaged dynamic pool space, calculate the value for the SYSGEN parameter NPAGEDYN using the following equation:

number = current value + total driver value + a + b + c + d

where:

current value Is the current byte count without any version of DECnet–VAX installed (derived from the SYSGEN command SHOW /MAJOR).

total driver value Is the total number of bytes required to load the driver plus NETDRIVER (see Table 5–3 for driver sizes).

a Is the total number of bytes required for all lines used by DECnet–VAX. Use the following formula to determine the approximate space required:

a = (number of lines) * (number of buffers) * (buffer size)

The buffer size and the number of buffers are determined by the values you assign to the executor BUFFER SIZE and individual line RECEIVE BUFFERS parameters.

b Is the total number of bytes required for the DECnet–VAX data structure that handles logical links. Use the following equation to determine the approximate space required:

b = 512 * number of links

The number of links is determined by the value you assign to the executor MAXIMUM LINKS parameter.

c Is the total number of bytes required for the DECnet data structure that handles circuits. Use the following equation to determine the approximate space required:

c = 60 * number of circuits

The number of circuits is the number of circuits you intend to use.

d Is a multiple of the additional IRPCOUNT count. Use the following equation to determine this value:

d = (new IRPCOUNT - current IRPCOUNT) * 200

The new IRPCOUNT is computed as in Section 5.4.1.2; the current IRPCOUNT is the current count without any version of DECnet–VAX installed.

Use the NCP SHOW EXECUTOR and SHOW CIRCUITS commands to display parameter values that you have set for the local node.

Table 5–3 provides the number of bytes required to load NETDRIVER and the individual drivers.

Table 5–3 Driver Sizes

Driver Software	Number of Bytes
CNDRIVER (CI)	4,000
ESDRIVER (SVA)	21,000
ETDRIVER (BNA)	25,000
NETDRIVER	18,000
NODRIVER (asynchronous)	15,000
SLDRIVER (DMB)	19,000
XDDRIVER (DMP)	11,500
XEDRIVER (UNA)	21,000
XGDRIVER (DMF)	16,500
XMDRIVER (DMC)	6,000
XQDRIVER (QNA)	21,000

5.4.1.2 IRPCOUNT Parameter

With the SYSGEN command SET IRPCOUNT, increase the number of preallocated I/O request packets to reflect the increase of nonpaged dynamic pool space, using the following formula:

number = current value + number of buffers + number of circuits + 2

where:

current value Is the current count without any version of DECnet–VAX installed.

number of buffers Is the number of buffers that may be used by DECnet–VAX in peak periods (see Section 5.4.1.1).

number of circuits Is the number of circuits that you intend to use.

5.4.1.3 LRPCOUNT and LRPSIZE Parameters

Calculate the value for the SYSGEN parameter LRPCOUNT according to the following formula:

number of buffers = number of lines + number of receive buffers

The number is the total number of receive buffers specified for all lines plus the total number of lines.

The value for LRPSIZE should match that of the BUFFER SIZE parameter in the executor database.

Refer to the *VMS System Generation Utility Manual* for a complete discussion of the System Generation Utility (SYSGEN). Note that the new settings for LRPCOUNT and LRPSIZE do not take effect until the next time the operating system is booted.

5.4.2 Critical Routing Node Requirements

For some critical routing nodes in large networks, you may need to guarantee that user processes running on the node never interfere with the memory requirements of the network software. In this case, you may want to configure the system for worst-case use of the nonpaged dynamic pool.

The use of nonpaged pool is controlled by the quota values that you specify for each user when you create the user's authorization record. For worst-case configuration, the sum of the quotas of all the simultaneously active processes must provide the required free pool for the network software.

To configure a system for the worst case, adhere to the following four rules:

- The sum of the ASTLM (asynchronous system trap limit), BIOLM (buffered I/O limit), DIOLM (direct I/O limit), and TQELM (timer queue limit) quotas for each process must be added to the IRPCOUNT parameter.
- The FILLM (file and logical link limit) quota for each process must be doubled and added to the IRPCOUNT parameter.
- The ENQLM (enqueue limit) quota for each process must be doubled and added to the IRPCOUNT parameter. ENQLM is the number of locks each process can own.
- The BYTLM (buffered I/O byte count limit) quota for each process must be added to the NPAGEDYN parameter.

Given these guidelines, the result of the following calculation should be less than the total free pool after the network has initialized:

```
USERPROCESSES * (((ASTLM + BIOLM + DIOLM + TQELM + (FILLM * 2)) * 96) + BYTLM)
```

For example, if for a critical routing node you also want to provide for 16 users' processes, the calculation might be as follows:

```
16 * (((10 + 6 + 6 + 5 + (8 * 2)) * 96) + 4096) = 16 * 8224 = 131584 bytes
```

This calculation indicates that there should be at least 131,584 bytes of nonpaged dynamic pool remaining after the network has initialized. However, because all users requiring their quota of pool at the same time is extremely unlikely, then, except for a worst-case configuration, the quotas and pool could be configured minimally so that no one user's quota would completely deplete the free pool. Use the SHOW MEMORY operator command or the dynamically updated POOL display for the Monitor Utility to configure the nonpaged dynamic memory pool for normal use.

The consequences of running almost or completely out of pool are fairly obvious to system users: System performance will be very sluggish; processes will continually enter the MWAIT scheduling state while they wait for an available free pool; and the free pool SHOW MEMORY display will indicate almost none.

If the lack of pool causes the network software on the node to be unable to allocate a buffer fast enough to receive data from a communications line, the line may be considered unusable by another node in the network. When this happens, the network attempts to adaptively reconfigure itself, thereby resulting in network traffic consisting of configuration update messages. If the node with pool problems is close to failing, without failing completely, it may alternate between working and not working, thereby causing the network to

repeatedly reconfigure itself. Ultimately, these reconfigurations degrade the performance of the entire network.

5.4.3 CPU Time Requirements

Compared to the procedures for configuring memory requirements and critical routing node requirements, proper system configuration to provide adequate processor time for the network software is somewhat more straightforward. Most of the procedures that control network routing are located in NETDRIVER. Because most of NETDRIVER runs at elevated interrupt priority level (IPL), normal user programs cannot preempt its execution. However, user-written drivers and privileged programs running at elevated IPL can affect the proper operation of NETDRIVER.

The *VMS Device Support Manual* provides guidelines for elevated IPL execution programming. In general, a program should run at elevated IPL only as long as necessary to synchronize correctly with other processes and devices. In particular, running at IPL IPL$_SYNCH for more than a few hundred milliseconds or running at any IPL at or above IPL$_MAILBOX for more than a few hundred milliseconds may adversely affect the network software. The effect of improper elevated IPL programming on the whole network is the same as having insufficient free nonpaged dynamic pool.

The NETACP process contains the procedure that handles the automatic network reconfiguration for a network node. Therefore, for proper network operation, the NETACP process must also be assured of sufficient processor time. It runs at a base priority of 9, which is well above the recommended base priority of 4 for normal users. However, real-time processes running at priorities 10 through 31 can preempt the execution of NETACP.

Just as for user-written drivers, the programming of real-time processes must account for the needs of the network software and other system software. A rule of thumb for real-time processes is that they should not normally preempt the execution of NETACP for more than a few hundred milliseconds at a time, and they should never preempt its execution for more than 5 to 10 seconds. This restriction allows NETACP sufficient processor time to run its routing algorithms properly.

If NETACP is unable to perform all of its functions, the effects on the whole network will be the same as having insufficient free pool or incorrect elevated IPL programming. If the preceding guidelines cannot be met for a particular real-time application, the application should probably not be used on a node that is also doing network routing.

The NETACP process, like all system processes, can be swapped and paged. However, because its base priority is 9, it is one of the last processes swapped when it is running and swapping becomes necessary. Also, NETACP receives high priority for paged disk I/O requests. Again, improper considerations for the disk I/O needs of NETACP can adversely affect the network as a whole. If the NETACP process continually enters the PFW or COMO scheduling state, it is probably not receiving sufficient priority for paging or swapping; other real-time or system programs should probably be modified to relieve the problem.

5.4.4 UNIBUS Adapter Map Register Considerations

The UNIBUS adapter on VAX processors provides a mechanism called UNIBUS map registers (UMRs) that allows UNIBUS peripherals to access VAX main memory. These map registers are required because the UNIBUS allows only 18 bits of address space while the VAX processors provide 24 or 30 bits (depending on processor type). The 18 bits of address space are divided into 496 pages of 512 bytes each. Therefore, there are 496 map registers that allow up to 496 pages of memory to be mapped for direct memory access (DMA) by UNIBUS devices.

Because UNIBUS map registers are a limited resource needed for correct operation of the network software, some consideration should be given to their use.

The VMS operating system provides a service for device drivers to allocate and deallocate map registers. Most drivers allocate map registers only for the time it takes for the device to perform a transfer, after which they deallocate them for use by another I/O request to the same device or by other drivers for different devices. This type of allocation is referred to as dynamic allocation.

Other drivers, however, permanently allocate map registers all the time the system is running. This is either because certain memory tables must be accessible to the device (TS11, DMC11, and DMP11) all the time the device is initialized or because of certain throughput requirements of the device (1 megabaud DMC11/DMR11, DMP11, DEUNA, and LPA11). This type of allocation is referred to as permanent allocation.

If a device driver process attempts to allocate UMRs and not enough are free to satisfy the allocation, the driver process is put into a FIFO wait queue to wait for UMRs to become available.

The UNIBUS map register requirements of the various supported UNIBUS devices are as follows:

- The TS11 driver permanently allocates a maximum of three map registers in its command table for each TS11 on the UNIBUS when it initializes. The TS11 supports one DMA I/O request at a time, and because an I/O can be up to 65,536 bytes in size, the TS11 driver requires a maximum of 128 map registers that can be allocated dynamically.
- The RL02, RK06, RK07, and RX02 disk drives support one DMA I/O request at a time. Because the maximum I/O request size can be 65,536 bytes, these drivers require a maximum of 128 map registers that can be allocated dynamically.
- The LPA11-K laboratory data acquisition device driver can be configured to permanently or dynamically allocate up to 496 map registers. The SYSGEN parameter LAMAPREGS allows you to specify how many map registers are to be permanently allocated; if this parameter is 0, map registers are dynamically allocated as needed when the device is used for I/O.
- The DMC11/DMR11 driver permanently allocates a maximum of three map registers for its base table when it initializes. It also permanently allocates enough map registers for all of its receive buffers, which you set with the RECEIVE BUFFERS line parameter and BUFFER SIZE executor parameter. However, if more than seven receive buffers are specified per line, only seven sets of map registers per line are allocated.

The DMC11/DMR11 driver supports one DMA transmit at a time and because each transmit can be a maximum of 16,383 bytes in size, the DMC11/DMR11 driver requires a maximum of 32 map registers that can be allocated dynamically.

- The DMP11 driver allocates map registers in the same way that the DMC11/DMR11 driver does with the exception that the DMP11 driver, which does not have a base table, requires three fewer map registers.
- The DEUNA device driver assigns eight receive buffers with a length of 1500 bytes per buffer. Enough map registers are permanently allocated for all receive buffers.

The following example details how one might determine whether sufficient map registers will be available. Assume the system consists of a VAX-11/780 with a single UNIBUS adapter, two RK07 disk drives on a single controller, a TS11 tape drive, and three DMR11s. The BUFFER SIZE executor parameter is set to 576 and each line has its RECEIVE BUFFERS parameter set to 4. Because each buffer can potentially cover three pages (two pages to contain the 576-byte buffer and one page for an offset page) and because the VMS map register allocator always allocates an extra invalid map register for protection, each buffer will require four map registers.

Device	Permanent Map Registers
TS11	Three permanent map registers would be required for the TS11.
DMR11(3)	Fifty-seven map registers would be required for the three DMR11s. (Each DMR permanently allocates 19 map registers; three UMRs for the base table and four UMRs for each of the four receive buffers.)

Therefore, 496 minus 60, or 436 map registers are available for dynamic use by all devices.

Because the TS11 and RK07s can each be using 128 map registers, there are always at least 436 minus 256, or 180 map registers available to map DMR11 transmit buffers. This amount is more than sufficient because the DMR11s each have only one transmit of 576 bytes outstanding at a time, which require a maximum of 3 times 4, or 12 map registers.

In general, because the VMS operating system has dynamic map register allocation and waiting when UNIBUS map registers run out, they are not a resource problem. However, if a system has a large number of DMC11/DMR11, DMP11, and DEUNA devices, you should calculate map register use to ensure that the configuration works.

5.4.5 Permanent Database Considerations in VAXclusters

The permanent configuration database, usually resident on disk, consists of a number of files. These files are listed in Table 5–4.

Table 5–4 Permanent Configuration Database Files

File Name	Usage
SYS$SYSTEM:NETNODE_REMOTE.DAT	Remote node
SYS$SYSTEM:NETNODE_LOCAL.DAT	Executor and loop node
SYS$SYSTEM:NETLINE.DAT	Line
SYS$SYSTEM:NETLOGING.DAT	Logging
SYS$SYSTEM:NETOBJECT.DAT	Object
SYS$SYSTEM:NETCIRC.DAT	Circuit
SYS$SYSTEM:NETX25.DAT	X.25 module
SYS$SYSTEM:NETX29.DAT	X.29 module
SYS$SYSTEM:NETCONF.DAT	Configurator module
SYS$SYSTEM:NETPROXY.DAT	Permanent proxy database

In a homogeneous VAXcluster, you may want to allow some of these files to be shared by members of the VAXcluster; shared files should be moved to SYS$COMMON:[SYSEXE] and files that are not to be shared should reside in SYS$SPECIFIC:[SYSEXE] (where they are normally created). Neither NETNODE_LOCAL.DAT nor NETX25.DAT should be shared because they contain executor information that is unique for each node in a VAXcluster. Other files such as NETLINE.DAT and NETCIRC.DAT should not be shared if the communications hardware configurations within the VAXcluster are not identical on every node.

As an example, if the permanent object database is identical on every node in a VAXcluster, you can make it shared by following these steps:

1 Rename (or move) the permanent object database on one node to the common system root, for example:

```
$ RENAME SYS$SPECIFIC:[SYSEXE]NETOBJECT.DAT -
_$ SYS$COMMON:[SYSEXE]NETOBJECT.DAT
```

or

```
$ COPY SYS$SPECIFIC:[SYSEXE]NETOBJECT.DAT -
_$ SYS$COMMON:[SYSEXE]NETOBJECT.DAT
```

2 Delete (or rename) the permanent object database from the private system root on each node in the VAXcluster, for example:

```
$ DELETE SYS$SPECIFIC:[SYSEXE]NETOBJECT.DAT;*
```

or

```
$ RENAME SYS$SPECIFIC:[SYSEXE]NETOBJECT.DAT;* -
_$ SYS$SPECIFIC:[SYSEXE]NETOBJECT.OLD;*
```

The files SYS$SYSTEM:NETNODE.DAT and SYS$SYSTEM:NETNODE_OLD.DAT are obsolete versions of the permanent node database that may be deleted.

6 Installation of a Network

This chapter describes how to start DECnet–VAX and how to install and start VAX PSI. Refer to the *Guide to DECnet–VAX Networking* for a summary description of the complete DECnet–VAX installation procedure.

A network consists of two or more nodes linked together. If there is no existing network to which you can connect your node, you can cooperate with the managers of other systems to create a new network. A new network is formed when two or more systems are connected by communications lines and each system is brought up as a network node.

The following sections describe how to register a DECnet–VAX key and how to bring up your node on a new or existing network.

6.1 Installing a DECnet–VAX Key

If you have purchased either a full function or an end node DECnet–VAX license and the appropriate full function or end node key, you must register the key on your system using the License Management Utility (LICENSE). The procedure for installing the DECnet–VAX key is described in the *VMS Version 5.0 Release Notes*. Refer to the *VMS License Management Utility Manual* for additional information on licensing.

If you have a DECnet–VAX full function license, registering the key allows you to configure your node as either a routing node or an end node. If you have an end node license, registering the key permits you to configure your node only as an end node. If you are upgrading from end node to full function capability, you must purchase an end node to full function license and register the DECnet–VAX key.

6.2 Bringing Up Your Network Node Using STARTNET.COM

After you satisfy the prerequisites for establishing a network and define the necessary parameters in the configuration database (see Chapter 5), you are ready to bring up your DECnet–VAX node. To do so, you must first define the local node (using the DEFINE EXECUTOR command) in the permanent database. This is the minimum requirement for the initial control of the operational state of the node.

After you build the permanent database using either NCP or the NETCONFIG.COM interactive configuration procedure (see Chapter 5), enter the following command to bring up your network:

```
$ @SYS$MANAGER:STARTNET.COM
```

This command starts NCP and NML, and configures the volatile database with the parameters that you defined in the permanent database. This procedure also turns on the local node and all lines and circuits connected to it. In addition, STARTNET.COM starts the network command terminal ACP by executing SYS$MANAGER:RTTLOAD.COM. At this point, the local node is ready for network operations with itself and with adjacent nodes.

DECnet–VAX uses OPCOM to display certain network-related messages on the network operator's console. When you turn on the local node, OPCOM displays the following message:

```
Opcom, hh:mm:ss:cc, SYSTEM Acct=
Opcom, DECnet starting
```

After you bring up your DECnet–VAX node, you use NCP commands to control the operational states of network components. You can control both local components and remote executions of NCP commands. This control allows you to dynamically reconfigure your network to control the use of the network and its resources. Use the NCP commands CLEAR and SET for the volatile database to control the network.

Parameters in the permanent database define network components each time you use the word ALL with the SET command. Typically, you use the SET "component" ALL command if you choose not to use STARTNET.COM to bring up the network. Section 6.5 discusses how to shut down the network.

Note that, if you are going to run VAX PSI, you must install the VAX PSI software before invoking STARTNET.COM. This also applies to VAX PSI Access software.

6.3 Bringing Up Your VAX PSI DTE

Bringing up VAX PSI DTE is similar to bringing up a node on the DECnet–VAX network. First, install VAX PSI, following the procedure described in the *VAX P.S.I. Installation Procedures*. Make sure you configure VAX PSI in multihost mode (instead of native mode) if you are building a DECnet network that allows some or all of its nodes to access the PSDNs connected. The multihost installation sets up the commands necessary to bring up the connector node (the node connected to the PSDNs) as a DTE. (Section 5.3 includes examples of both native mode and multihost mode configurations.)

Invoking the STARTNET.COM procedure brings up both DECnet–VAX and VAX PSI software. After VAX PSI is installed properly, you can use VAX PSI to communicate with a remote DTE over a packet switching data network (PSDN).

Note that if you make changes to your PSI system while it is running and you want to include these changes in your PSI system the next time the system is started, you must also make similar changes to the permanent database using NCP commands (DEFINE and PURGE).

6.4 Testing the Installation with UETP Test Procedure

To ensure that the DECnet–VAX installation is successful, you can use the User Environment Test Package (UETP) to test DECnet. The test procedure is described in the *Guide to Setting Up a VMS System*.

6.5 Shutting Down Your DECnet–VAX Node

Bringing down your operating system automatically brings down your DECnet–VAX node as well. The next time you reboot the operating system, your network comes up automatically if SYSTARTUP_V5.COM invokes STARTNET.COM (see Section 6.2). However, if the network is running and you want to shut down your network node in an orderly manner or otherwise restrict its use, you can use NCP to control the operational state of the local node. NCP offers three options for shutting down the executor node.

- To shut down your local node without destroying active logical links, enter the following command:

  ```
  NCP>SET EXECUTOR STATE SHUT
  ```

  This command closes the node in an orderly fashion; new links are not allowed, and existing links are not destroyed. When all logical links are disconnected, this command turns off the node, and NCP logs an event message.

  When the last link terminates and is disconnected, the executor node in the SHUT state enters the OFF state. This action occurs whether or not the node is currently in use for route-through traffic. Consequently, the communication path between nodes using the local node for route-through may be broken.

- Instead of shutting down your local node, you can restrict network operations on that node. This restriction does not affect current logical link activity; however, no new inbound logical links can be created unless they originate locally or unless a process with the OPER privilege confirms them. Enter the following command to restrict local node operations:

  ```
  NCP>SET EXECUTOR STATE RESTRICTED
  ```

- To shut down the local node regardless of current logical link activity, enter the following command:

  ```
  NCP>SET EXECUTOR STATE OFF
  ```

  This state allows no new logical links to be created, terminates existing links, and stops route-through traffic.

Note: Programs that have declared names or object numbers and that are started independently of DECnet-VAX should be programmed to terminate when their mailboxes receive a MSG$_NETSHUT message. This message appears when the node is shutting down.

Whenever the local node's state goes to OFF, DECnet–VAX uses the OPCOM facility to display the following message on the console:

```
Opcom, hh:mm:ss:cc, SYSTEM Acct=
Opcom, DECnet shutting down
```

Table 6–1 summarizes local node states and basic network operation restrictions for them. These operations include network routing, confirming inbound connections from a remote node, and initiating outbound connections to a remote node.

6.5 Shutting Down Your DECnet–VAX Node

If your network configuration includes VAX PSI, shutting down a DECnet–VAX node also shuts down VAX PSI and clears the PSI volatile database. Bringing up the DECnet–VAX node subsequently restarts VAX PSI and re-creates the volatile database from the permanent database.

Table 6–1 Local Node States and Network Operations

State	Route-Through Traffic	Connect Confirm Operations	Connect Initiate Operations
ON	Unrestricted	Unrestricted	Unrestricted
RESTRICTED	Unrestricted	Unrestricted only if the partner node is the local node or if the confirming process has the OPER privilege	Unrestricted
SHUT	Unrestricted	Unrestricted only if the confirming process has the OPER privilege	Unrestricted only if the initiating process has the OPER privilege
OFF	Restricted	Restricted	Restricted

7 Testing the Network

NCP provides several kinds of tests to help you determine whether the network is operating properly. Specifically, these tests let you exercise network software and hardware by sending data through various network components and then returning that data to its source. After you start DECnet–VAX software, you may want to run some of these tests.

In general, problems that you encounter with the DECnet–VAX network probably arise from misconfigured VMS and DECnet parameters that you can fix using SYSGEN or NCP. DIGITAL supplies variations of these tests to exercise separate layers of the network. User-written processes or DECnet-supplied processes may also initiate the tests.

DECnet–VAX tests fall into two categories: node-level loopback tests and circuit-level loopback tests. Use node-level tests to evaluate the operation of logical links, routing, and other network-related software. Use circuit-level tests to evaluate the operation of circuits. Using node-level tests first is recommended; then, if necessary, use circuit-level tests. This chapter describes these variations as they relate to DECnet loopback capabilities and the NCP command LOOP, and provides a practical approach to their use. Note that you cannot use LOOP commands on asynchronous lines or circuits.

VAX PSI provides various ways to analyze software and hardware operation and to diagnose problems in PSI operations. Use line-level loopback tests to evaluate the operation of the X.25 physical lines and communications hardware. VAX PSI provides an additional facility for the KMS11 and KMV11 line interfaces, allowing you to dump the microcode to a specified file for analysis. This chapter describes these VAX PSI test facilities and how to use them. For additional details about the VAX PSI test facilities, refer to the *VAX P.S.I. Problem Solving Guide.*

7.1 Node-Level Tests

Node-level loopback tests examine the logical link capabilities of a node by exchanging test data between DECnet processes on two different nodes or between DECnet processes on the same node. There are two types of test:

- Loopback tests for logical link operation regardless of the circuit
- Loopback tests for operation over a specified circuit

The second test sends test messages over a specified circuit associated with a loop node name (see Section 7.1.2). This test directs test messages regardless of the Routing layer function.

Both types of node-level loopback test allow you to test the functions of your DECnet–VAX software. To test various aspects of this software, you may want to perform a series of operations, as follows:

1 In the first test, loop information to a remote loopback mirror process using a remote loopback test. This tests all local and remote network software up to the DNA user layer on the remote node.

2 If the first test fails, use a loop node name and loop information to the local node and to a remote node. The loop node name allows you to direct traffic over a specified circuit, which tests local and remote Routing layer software.

3 If the second test fails, set the circuit's line to "controller loopback" and repeat step 2.

Regardless of the type of test you choose, use the NCP command LOOP NODE to send test messages. This NCP function uses a cooperating process called the Loopback Mirror to facilitate the transmission and reception of test messages. When you use this command, you have the option of controlling the type of binary information (MIXED, ONES, ZEROS); the number of blocks of information, which ranges from 1 to 65,535; and the length in bytes of each block to be looped, which also ranges from 1 to 65,535. (Using a maximum block length of 4096 bytes to reduce the system load is recommended.) Refer to the *VMS Network Control Program Manual* for the complete syntax of the LOOP NODE command.

If your message returns with an error, the test stops and NCP prints a message that indicates a test failure, specifies the reason for the failure, and provides a count of the messages that were not returned. For a summary of NCP error messages, refer to the *VMS System Messages and Recovery Procedures Reference Volume*.

In the following example, the test attempts to send 10 messages, each 50 bytes long. The first two messages are sent successfully, and an error occurs on the third.

```
NCP>LOOP NODE BOSTON COUNT 10

%NCP-W-LINCOM, line communication error

Messages not looped = 8
```

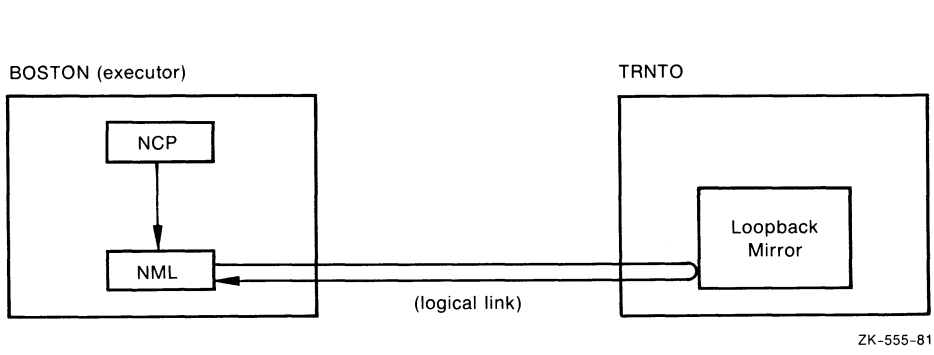
7.1.1 Remote Loopback Test

Use the LOOP NODE command to test for a logical link connection between two nodes. When using this command, you must identify the node to which you want to loop test messages. Figure 7–1 illustrates a remote loopback test.

For this test, you first turn the selected remote node line and circuit to the ON state to allow for logical link activity. Then, you use the LOOP NODE command. For example, the following set of commands tests both local and remote DECnet software on nodes BOSTON and TRNTO:

```
NCP>SET LINE DMC-0 STATE ON
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>LOOP NODE TRNTO COUNT 10
```

Figure 7–1 Remote Loopback Test



7.1.2 Local and Remote Loopback Tests Using a Loop Node Name

If the remote loopback test fails, then use the LOOP NODE command with a loop node name to test a logical link path over a specified circuit. You can loop test messages either over a logical link path and circuit within the local node or between two different nodes with a **loop node** specified for the circuit to be used. Use the latter method first in order to test remote Routing layer software. In each case, use the SET NODE command with the CIRCUIT parameter to establish a loop node name. For example, the following command establishes circuit DMC–0 as the circuit over which loop testing will take place:

```
NCP>SET NODE TESTER CIRCUIT DMC-0
```

No other parameters are valid for loop nodes. This circuit must be turned on when performing these tests.

Note that you cannot assign two loop node names to the same circuit. For example, after you establish TESTER as the loop node name for circuit DMC–0, you must enter a CLEAR NODE TESTER CIRCUIT command before assigning another loop node name to DMC–0.

When a logical link connection request is made to the loop node name, all subsequent logical link traffic is directed over the associated circuit. The destination of the traffic is whatever node address is associated with the loop node name. The loop node name is necessary because, under normal operation, DECnet Routing software selects which path to use when routing information. The loop node name overrides the routing function so that information can be routed over a specific circuit. To remove the association of the loop node name with a circuit, use the CLEAR NODE CIRCUIT or CLEAR NODE ALL command, as in the following example:

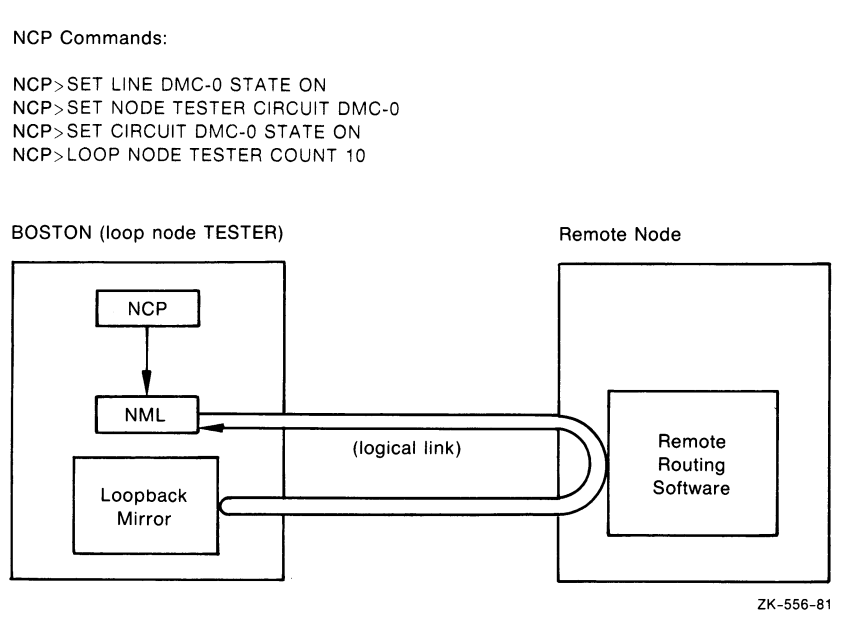
```
NCP>CLEAR NODE TESTER CIRCUIT
```

A loop node name specified with the SET NODE CIRCUIT command may be used for any network traffic (for example, COPY requests or application program traffic). The loopback node name appears as a valid node name in the network for all purposes.

7.1.2.1 Local-to-Remote Testing

To test a logical link path over a circuit between the local node and a remote node, you must specify a loop node name for the given circuit and enter the LOOP NODE command. Figure 7–2 illustrates a local-to-remote loopback test using a loop node name.

Figure 7–2 Local-to-Remote Loopback Test Using a Loop Node Name



For this test, you first turn on the line and set a loop node name for the given circuit to the remote node. Next, turn on the circuit. Finally, enter the LOOP NODE command using the loop node name, as shown in the following example:

```
NCP>SET LINE DMC-0 STATE ON
NCP>SET NODE TESTER CIRCUIT DMC-0
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>LOOP NODE TESTER COUNT 10
```

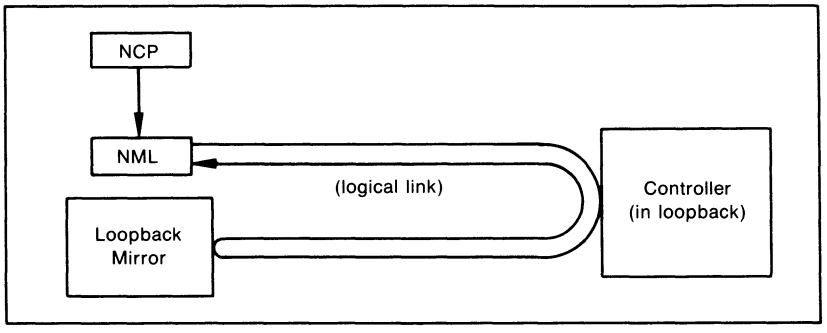
This set of commands tests both local and remote Routing layer software operation. The test messages are looped over the loopback circuit. Because the test actually tests the operation of the Routing layer on the remote node, the message may not come back on the circuit over which it was sent.

Figure 7–3 Local-to-Local Loopback Test Using a Loop Node Name

NCP Commands:

```
NCP>SET LINE DMC-0 STATE OFF
NCP>SET LINE DMC-0 CONTROLLER LOOPBACK
NCP>SET LINE DMC-0 STATE ON
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>SET NODE TESTER CIRCUIT DMC-0
NCP>LOOP NODE TESTER COUNT 10 LENGTH 32
```

BOSTON (loop node TESTER)



ZK-557-81

7.1.2.2 Local-to-Local Testing

If the local-to-remote test fails, try a local loopback test with the local node to test local Routing layer software exclusively. To test a logical link path over a specified line on the local node, specify a loop node name and set the device controller to loopback mode. Figure 7–3 illustrates a local-to-local loopback test using a loop node name.

For this test, you first turn off the line, set the controller to loopback mode, and turn on the line and circuit. Finally, set a loop node name for the given line and enter the LOOP NODE command using the loop node name. The following set of commands tests the Routing layer software and the controller on the local node:

```
NCP>SET LINE DMC-0 STATE OFF
NCP>SET LINE DMC-0 CONTROLLER LOOPBACK
NCP>SET LINE DMC-0 STATE ON
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>SET NODE TESTER CIRCUIT DMC-0
NCP>LOOP NODE TESTER COUNT 10 LENGTH 32
```

Because the device is set to loopback mode, the test messages are looped over the circuit and back to the local node. If this test fails, try a local loopback test to test local DECnet software.

Note: Because of restrictions in the operation of the DMC controller, you must use a block length of fewer than 50 bytes for controller loopback tests.

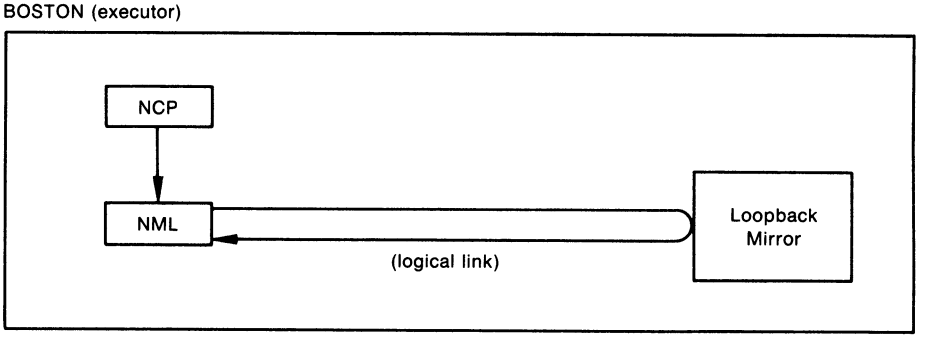
7.1.3 Local Loopback Test

If the loopback tests described in Section 7.1.2.2 fail, then use either the LOOP NODE command with the local node-id or the LOOP EXECUTOR command to test local DECnet software. This type of test uses DECnet–VAX software to loop messages to the loopback mirror on the local node. Figure 7–4 illustrates a local loopback test.

Figure 7–4 Local Loopback Test

NCP Command:

```
NCP>LOOP EXECUTOR COUNT 10
```



For this test, you enter the following command at the local node:

```
NCP>LOOP EXECUTOR COUNT 10
```

This test evaluates the local DECnet software using an internal logical link path. If this test succeeds and the other node-level tests fail, then try the circuit-level tests. If these tests fail, the executor's default nonprivileged DECnet account is probably set up incorrectly.

7.2 Circuit-Level Tests

Circuit-level loopback tests examine a DECnet circuit by looping test data through a hardware loopback device on the circuit, either through a modem (or loopback connector) or through a remote node. The tests that use a hardware loopback device are referred to as controller loopback tests. The tests that use a loopback connector or a modem are referred to as circuit loopback tests. The tests that use the software capabilities of the system are referred to as software loopback tests.

You may want to perform a series of operations to test various aspects of a circuit, as follows:

1 In the first test, perform a software loopback test to another node to determine whether the circuit is operational up to the remote circuit unit and controller.

2 If the first test fails, set the controller to loopback mode and use a controller loopback test to determine whether the controller works.

3 If the second test succeeds, then attach a modem (or loopback connector) to the controller and use a circuit loopback test to determine whether the unit is functional.

Regardless of the test type, you must use the NCP command LOOP CIRCUIT to perform a circuit-level loopback test. When you enter this command, you have the option of controlling the type of binary information (MIXED, ONES, ZEROS); the number of blocks of information, which ranges from 1 to 65,535; and the length in bytes of each block to be looped, which also ranges from 1 to 65,535. (Using a maximum block length of 4096 bytes is recommended.) For the complete syntax of the LOOP CIRCUIT command, refer to the *VMS Network Control Program Manual.*

If your message returns with an error, the test stops and NCP issues a message indicating a test failure, the reason for the failure, and a count of the messages that were not returned. For a summary of NCP error messages, refer to the *VMS System Messages and Recovery Procedures Reference Volume.* In the following example, the test attempts to send 10 messages, each 50 bytes long. The first two messages are sent successfully, and an error occurs on the third.

```
NCP>SET LINE DMC-0 CONTROLLER NORMAL STATE ON
NCP>LOOP CIRCUIT DMC-0 COUNT 10

%NCP-W-LINPRO, line protocol error

Messages not looped = 8
```

7.2.1 Software Loopback Test

Use the LOOP CIRCUIT command to perform a software loopback test of a circuit connected to the local node. This type of test uses DECnet–VAX software to loop through the circuit-to-circuit service software in the adjacent node and back to the local node. Figure 7–5 illustrates a software loopback test that checks whether the circuit is operational up to the remote unit and controller on the adjacent node.

In the first step of this test, you turn off the line. Next, you set the controller to its normal operational mode and put the line and the circuit in the ON state. Finally, you enter the LOOP CIRCUIT command, as shown in the following example:

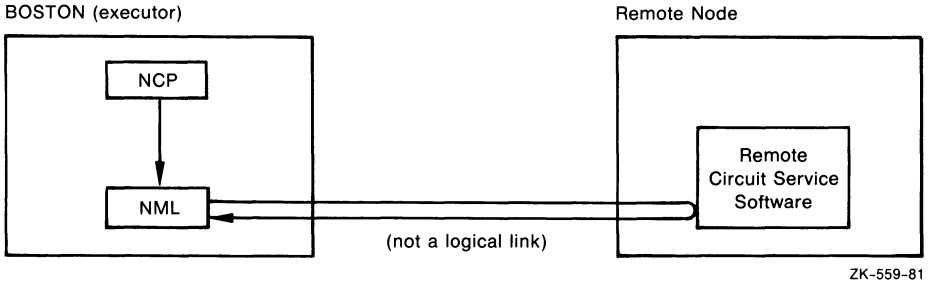
```
NCP>SET LINE DMC-0 STATE OFF
NCP>SET LINE DMC-0 CONTROLLER NORMAL
NCP>SET LINE DMC-0 STATE ON
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>LOOP CIRCUIT DMC-0 COUNT 10
```

This set of commands tests the circuit DMC–0 up to the adjacent node. If this test fails, try a circuit loopback test to verify that the circuit is functional.

Figure 7–5 Software Loopback Test

NCP Commands:

```
NCP>SET LINE DMC-0 STATE OFF
NCP>SET LINE DMC-0 CONTROLLER NORMAL
NCP>SET LINE DMC-0 STATE ON
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>LOOP CIRCUIT DMC-0 COUNT 10
```



7.2.2 Controller Loopback Test

Use the LOOP CIRCUIT command to perform a controller loopback test of a physical line on the local node while the controller is in loopback mode. This type of test verifies whether the circuit up to the controller and the controller itself are functional. Figure 7–6 illustrates a controller loopback test.

For this test, you first turn off the line. Next, you set the controller to loopback mode and put the line and circuit in the ON state. Finally, you enter the LOOP CIRCUIT command. For example:

```
NCP>SET LINE DMC-0 STATE OFF
NCP>SET LINE DMC-0 CONTROLLER LOOPBACK
NCP>SET LINE DMC-0 STATE ON
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>LOOP CIRCUIT DMC-0 COUNT 10 LENGTH 32
```

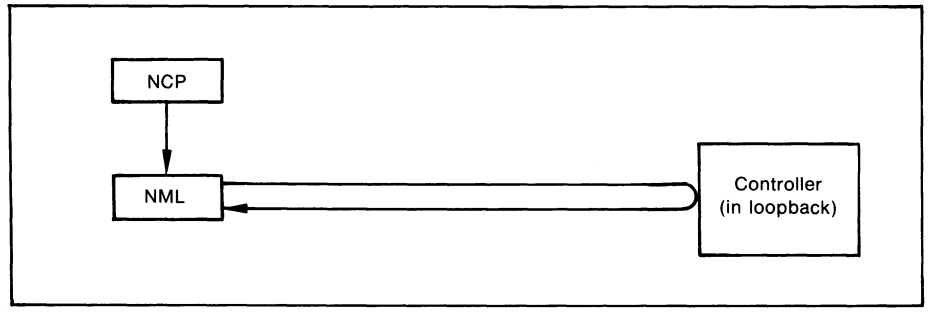
This set of commands tests the circuit up to the controller for physical line DMC–0 connected to the local node by circuit DMC–0.

Note: Because of restrictions in the operation of the DMC controller, you must use a block length of fewer than 50 bytes for controller loopback tests.

Figure 7–6 Controller Loopback Testing

NCP Commands:

```
NCP>SET LINE DMC-0 STATE OFF
NCP>SET LINE DMC-0 CONTROLLER LOOPBACK
NCP>SET LINE DMC-0 STATE ON
NCP>SET CIRCUIT DMC-0 STATE ON
NCP>LOOP CIRCUIT DMC-0 COUNT 10 LENGTH 32
```



7.2.3 Circuit-Level Loopback Testing

Circuit-level loopback testing is also supported for Ethernet circuits. One major difference between loopback testing on point-to-point and multipoint circuits (DMCs and DMPs) and on an Ethernet circuit is that the former requires two separate processors (one at each end), but the latter requires only one processor. In Ethernet circuit loopback testing, the target node's Ethernet interface, rather than its processor, loops the messages.

In Ethernet circuit-level loopback testing (as in the case of point-to-point circuit loopback testing), network management accesses the Data Link layer directly, thus bypassing intermediate layers. One advantage of the Ethernet loopback test is that it can be performed concurrently with other DECnet operations on the circuit.

7.2.3.1 Testing with the PHYSICAL ADDRESS and NODE Parameters

To be tested, an Ethernet circuit must be in the ON state and the SERVICE parameter must be set to ENABLED. Note that, by default, the SERVICE parameter is set to DISABLED for Ethernet circuits. As indicated in Chapter 2, DECnet supports the UNA, which provides for multiaccess connections between many nodes on the same Ethernet circuit. In the following example, the command identifies the circuit device UNA and the controller number 0 for an Ethernet circuit:

```
NCP>SET CIRCUIT UNA-0 STATE ON SERVICE ENABLED
```

The UNA is used to loop messages on the Ethernet circuit. If desired, it can be used to loop messages to itself in order to test its own state. To do this, enter the following commands:

```
NCP>SET LINE UNA-0 STATE OFF
NCP>SET LINE UNA-0 CONTROLLER LOOPBACK STATE ON
NCP>SET NODE TEST CIRCUIT UNA-0
NCP>LOOP NODE TEST
```

In this case, you are able to test the status of the UNA in controller loopback, but not the capacity of the node to transmit and receive messages. For more information about the node's capacity to send and receive messages, see Section 7.1.2.2.

More typical cases of loopback testing of Ethernet circuits involve looping messages to remote systems over the Ethernet; this tests the capability of both the local and the remote UNAs to send and receive messages. In those cases, you are required to supply such information as the Ethernet physical address or the node name or address of the circuit at the remote node that you want to test.

Nodes on Ethernet circuits are identified by unique Ethernet addresses. An Ethernet address is 48 bits in length and is represented by six pairs of hexadecimal digits (6 bytes), separated by hyphens (for example, AA-01-23-45-67-89). For more detail on Ethernet addresses, see Section 3.3.4.

Each UNA on the Ethernet circuit has a hardware address (in read-only memory) that has been assigned to it by the manufacturer. Typically, DECnet sets an Ethernet physical address for the UNA, thereby replacing the hardware address as the address to which the UNA currently responds. The UNA's physical address continues to be the address to which it responds, unless it is reset to the hardware address value (for example, if the Ethernet circuit is set to OFF).

Knowing the Ethernet physical address of the UNA on the remote node that you want to test is helpful. Because this is not always possible, you should plan to perform loopback tests to include the hardware address of each of the UNAs on your Ethernet circuit in the permanent database, thus ensuring that the address is retrievable from the volatile database. You can then use the node-id in the LOOP command. When you specify node-id, the network management software retrieves the hardware address from the volatile database and attempts to transmit the loop message to the remote UNA by alternately using the hardware address and the physical address that DECnet normally uses.

The following example contains an Ethernet physical address:

```
NCP>LOOP CIRCUIT UNA-0 PHYSICAL ADDRESS AA-00-04-00-FF-04
```

Because, in this case, you know the physical address of the remote node that you want to test, you merely include the PHYSICAL ADDRESS parameter with its value. If, however, that physical address had changed (for example, if it had been reset to the hardware address value), the loopback would have failed. You would have received the following message:

```
%NCP-W-LINPRO, line protocol error
Messages not looped = 8
```

If you also know the name or address of the remote node, you could test the UNA on that node even though its Ethernet physical address may have changed. The Ethernet hardware address of the node to be tested must already have been entered in the database on the executor node. If the hardware address is included in the volatile database, and you test by supplying the node name or address, the loop test is attempted by the network management software to both the hardware address and the DECnet address.

An example of a loopback test that specifies the NODE parameter is the following:

```
NCP>LOOP CIRCUIT UNA-0 NODE TEST
```

Assume that TEST's physical address, which was AA–00–04–00–F7–04, is changed. Thus, any attempt to test TEST using the old physical address does not succeed. If, however, TEST's hardware address (which was AA–00–03–00–01–31) is included in the volatile database on the executor node, the loopback test with the NODE parameter in its specification does succeed.

In the preceding example, you could alternatively supply the node address value (such as 226) for the NODE parameter. For example, if you know the node-id but not the name of the node, you could enter the following:

```
NCP>LOOP CIRCUIT UNA-0 NODE 226
```

In this case, the node address is used to construct the DECnet physical address, and the Ethernet hardware address (assuming that it is included in the volatile database) is used to access the circuit on the remote node and complete the loopback test. Thus, entering the hardware address in the volatile database is important.

If you want to examine the Ethernet hardware address of your own UNA (in this case UNA–0), you can use the NCP command SHOW LINE CHARACTERISTICS. For example:

```
NCP>SHOW LINE UNA-0 CHARACTERISTICS
```

When you enter this command, you receive a display such as the following:

```
Line Volatile Characteristics as of 30-DEC-1988 15:33:25

Line = UNA-0

Receive buffers         = 0
Controller              = normal
Protocol                = Ethernet
Service timer           = 4000
Hardware address        = AA-00-03-00-12-00
Buffer size             = 1498
```

7.2.3.2 Loopback Assistance

DECnet supports the use of an assistant physical address and an assistant node to aid you in interrogating a remote node. To use this feature, you specify either the ASSISTANT PHYSICAL ADDRESS parameter or the ASSISTANT NODE parameter as an additional parameter to the LOOP CIRCUIT command.

You can use the "assistant" in three distinct ways. First, you can use it to assist you in receiving loop messages from a remote node. Second, you can use it in transmitting loop messages to a remote node. Third, you can use it in both transmitting messages to and receiving messages from a remote node.

There are various reasons why you might choose one form of assistance over another. For example, the target node to which you want to transmit a message may be located at a point where the signals are too weak to send a message. In this case, you could request assistance in transmitting the message to the target node. Similarly, you may be able to transmit messages to the target node, but not be able to receive messages from it. In such a case you can send a message directly to the target node and request an "assistant" to aid you in receiving a message from the target node. When you encounter difficulties in both sending and receiving messages, you can request an assistant node to help you to both transmit messages to and receive messages from the target node.

The following commands illustrate how to use the ASSISTANT PHYSICAL ADDRESS and ASSISTANT NODE parameters:

```
NCP>LOOP CIRCUIT UNA-0 PHYSICAL ADDRESS AA-00-04-00-18-04 -
  ASSISTANT PHYSICAL ADDRESS AA-00-04-00-15-04

NCP>LOOP CIRCUIT UNA-0 NODE LOON ASSISTANT NODE THRUSH
```

In the first command, you are requesting the node described by the Ethernet physical address AA–00–04–00–15–04 to assist you in testing the node described by the Ethernet physical address AA–00–04–00–18–04. In the second command, you are requesting the node THRUSH to assist you in testing node LOON.

If you specify either the ASSISTANT PHYSICAL ADDRESS or ASSISTANT NODE parameter and you do not specify the HELP parameter, you receive FULL assistance; that is, you are assisted both in receiving and transmitting loop messages. Note that, in the preceding examples, because the ASSISTANT PHYSICAL ADDRESS and ASSISTANT NODE parameters are specified without the HELP parameter, the default is FULL assistance.

If you want to use an assistant node only to receive messages from the remote node, you could enter the following command:

```
NCP>LOOP CIRCUIT UNA-0 NODE LOON ASSISTANT NODE THRUSH HELP RECEIVE
```

In this example you are requesting the node THRUSH to assist you in receiving messages from node LOON. When you want to be assisted only in sending or transmitting loop messages, you could enter a command such as the following:

```
NCP>LOOP CIRCUIT UNA-0 NODE LOON ASSISTANT NODE 21 HELP TRANSMIT
```

Note that, in this case, the ASSISTANT NODE parameter contains the node address, rather than the name of the node as in the previous example. In each of the last two examples, the HELP parameter is included to specify the type of assistance desired.

7.3 X.25 Line-Level Loopback Tests

There are three types of line-level loopback tests that you can use to test an X.25 physical line:

- External loopback tests that loop data back through the modem
- Internal loopback tests that loop data back through the device
- External loopback tests that loop data back through a loopback device on the line

You may want to perform the following series of operations to test various aspects of the physical line:

1. First, using the loop switch on the modem, perform an external loopback test through the modem. This test checks the logic of the device transmitter and receiver, the line driver, the modem cable, and part of the modem. If this test is successful and you still have errors, contact your network manager.
2. If the first test fails, perform an internal loopback to test only the logic of the device transmitter and receiver.
3. If the second test succeeds, attach a hardware loopback device to the modem cable. Then perform an external loopback to test the logic of the device transmitter and receiver, the line driver, and the modem cable.

Regardless of the test type, use the NCP command SET LINE to specify the type of loopback test and the NCP command LOOP LINE to initiate the line-level loopback test.

Specify values for two parameters of the SET LINE command, as follows:

Internal Loopback	External Loopback
STATE SERVICE	STATE SERVICE
CONTROLLER LOOPBACK	CONTROLLER NORMAL

Note that the line state must be set to OFF before the CONTROLLER parameter can be changed.

To initiate a test, use the LOOP LINE command with the same line identifier you specified with the SET LINE command. For example, the following commands initiate an external loopback test for the line DUP-0:

```
NCP>SET LINE DUP-0 ... STATE SERVICE CONTROLLER NORMAL
NCP>LOOP LINE DUP-0 ...
```

Associate parameters with the LOOP LINE command to control the type of test information and the size and number of blocks sent during testing.

Use the COUNT and LENGTH parameters to specify the number of blocks sent over the line during a test and the length (in bytes) of each block sent. The following command sends 2000 blocks 100 bytes long over the line:

```
NCP>LOOP LINE DUP-0 COUNT 2000 LENGTH 100 ...
```

Specify decimal integers in the range 1 to 65,535 for both these parameters. Note that this test takes approximately 5 minutes, as the following calculation shows:

```
2000 * (100+4)*8
---------------- seconds  = 5 minutes
       5000
```

A DUP runs at 5000 bps when looped back.

Use the WITH parameter to specify the type of binary information sent during loopback testing. You can specify three types of binary information:

ONES	All binary 1s
ZEROES	All binary 0s
MIXED	A random combination of 1s and 0s

For example, the following command sends 2000 blocks 100 bytes long, each containing all binary 1s, over the line:

```
NCP>LOOP LINE DUP-0 COUNT 2000 LENGTH 100 WITH ONES
```

If you omit the WITH parameter, a combination of 1s and 0s (MIXED) is sent. If you omit the COUNT and LENGTH parameters, one block of 128 bytes is sent. For example, the following command sends one block of 128 bytes, containing mixed binary information, over the line:

```
NCP>LOOP LINE DUP-0
```

The *VAX P.S.I. Problem Solving Guide* provides further details on loopback testing.

7.4 Dumping KMS11 and KMV11 Microcode

This section describes how to dump the KMS11 or KMV11 microcode to a file and how to analyze the dump file. The KMS11 and KMV11 are synchronous line devices that interpret the X.25 level 2 protocol. They are supported by VAX PSI (see the table of DECnet circuit and line devices in the *VMS Network Control Program Manual*).

Use the MICROCODE DUMP parameter of the NCP command SET LINE to dump the microcode of the specified device to the file indicated. By default, the output file takes the following format:

SYS$ERRORLOG:filename.DMP

where:

filename Is the file you specify.

For example, the following command dumps the microcode of the file BARRY.DMP in the SYS$ERRORLOG directory:

```
NCP>SET LINE KMX-0-0 MICROCODE DUMP BARRY
```

You can use the DUMP parameter only if you believe there is an error in the microcode of the KMX, KMY, or KMV.

You can use KMS/KMV dump analyzer to process the dump file. For a description of the dump analyzer, refer to the *VAX P.S.I. Problem Solving Guide*.

Part IV Network User Operations

8 Performing Network User Operations

DECnet–VAX allows you to perform a variety of operations over the network:

- Retrieve information about the status of the nodes in your network.
- Establish communication with a remote DECnet node through the heterogeneous command terminal facility.
- Access files on remote nodes.
- Perform task-to-task operations.

This chapter describes each of these operations. The primary focus of this chapter, however, is on the use of task-to-task communication in network operations.

For VAX PSI user operations, refer to the VAX PSI documentation set.

8.1 Retrieving Network Status Information

Before you perform a specific type of operation over the network, you may want to check the status or availability of a particular node or nodes in your network. To retrieve such information, you can use the DCL command SHOW NETWORK. The SHOW NETWORK command displays the availability of the local node as a member of the network.

Note that you can use the SHOW NETWORK command to retrieve information about other nodes in your network only if your local node is a routing node. If your local node is a nonrouting (end) node, you do not receive any network information; instead, you are directed to a designated routing node. If your node is an area router, the SHOW NETWORK command displays additional information about the area.

The SHOW NETWORK command also displays link and cost relationships between the local node and other nodes in the network. It displays the following characteristics about the current network:

Node	Identifies each available node in the network by its node address and node name.
Links	Shows the number of logical links between the local node and each available remote node.
Cost	Shows the total line cost of the path to a remote node. The system manager assigns the cost for each line in the network.
Hops	Shows the number of intervening nodes plus the target node.
Next hop to node	Shows the outgoing physical line used to reach the remote node. (The local node is identified by the term LOCAL.)

Area	Identifies each available area in the network by its area number. This characteristic is displayed only if the local node is an area router.
Next hop to area	Shows the outgoing physical line used to reach the remote area. This characteristic is displayed only if the local node is an area router. The local node is identified by the term LOCAL. The node address and node name of the next hop to the target area are also displayed.

When you enter the SHOW NETWORK command on a level 1 router (a router that is not an area router), you receive a display on your terminal similar in format to the following:

```
VAX/VMS Network Status for local node 2.1 NYC on 30-DEC-1988 09:18:03.07

The next hop to the nearest area router is node 2.62 ZEUS.

    Node        Links  Cost   Hops  Next Hop to Node

2.1     NYC       0     0      0     Local   ->   2.1    NYC
2.2     RAEL      0     8      1     UNA-0   ->   2.2    RAEL
2.3     PANGEA    0     8      1     UNA-0   ->   2.3    PANGEA
2.4     TWDEE     0    10      2     UNA-0   ->   2.63   AURORA
2.5     TWDUM     0     8      1     UNA-0   ->   2.5    TWDUM
2.11    NEONV     0     8      1     UNA-0   ->   2.11   NEONV
2.63    AURORA    0     8      1     UNA-0   ->   2.63   AURORA

            Total of 7 nodes.
```

If your local node is an end node, and you enter the SHOW NETWORK command, you receive the following message on your terminal:

```
This is a nonrouting node, and does not have any network information.
The designated router for node NYC is node 2.62 ZEUS.
```

If you enter the SHOW NETWORK command, but the network is unavailable at that time, you receive the following display:

```
Network unavailable
```

For more detailed information about the DCL command SHOW NETWORK, see its description in the *VMS DCL Dictionary*.

8.2 Establishing Communication with a Remote Node

DECnet–VAX supports a command terminal facility that permits users to establish communication with a remote node and to use the facilities of that system while physically connected to the local node. By means of this link, you can temporarily become a local user of the remote node and thereby perform functions that the remote node allows its local users to perform from a terminal.

Note that, in addition to communicating with remote VMS nodes, you can communicate with non–VMS nodes that support the DNA heterogeneous remote command terminal protocol facility (also referred to as the network virtual terminal facility). Consult the Software Product Description for a description of non–VMS operating systems and their DECnet implementations.

8.2 Establishing Communication with a Remote Node

If you want to use the command terminal facility to establish communication with a remote node, enter the DCL command SET HOST in the following format:

```
$ SET HOST nodename
```

where:

nodename Is a 1- to 6-character name or number specifying the remote node at which you want to log in.

The SET HOST command does not recognize the area prefix in a node number. Therefore, to specify by number a node in another area, you must convert the node number to its decimal equivalent, as described in Section 3.7.2.

The operating system on the remote node prompts for a user name and password. If the information you supply is valid, you are logged in to the remote node. To return control to your local node, type LOGOUT.

If the remote node is a VMS node, you receive the following message at your terminal after you type LOGOUT:

```
%REM-S-END, control returned to node _NODENAME::
```

This message indicates that control is returned to your local node.

The only special control character used for remote command terminal operations is CTRL/Y. Except for CTRL/Y, all control characters are handled as if they were issued at the local node.

Repeated, rapid pressing of CTRL/Y generates a prompt asking if the remote connection should be broken. If you answer YES to the prompt, control returns to the local node. This technique is useful if for some reason you cannot return to the local node normally.

The following command sequence illustrates the operation of remote command terminals for the network topology example. The name of the local node is BOSTON.

```
$ SET HOST TRNTO
Username: SMITH
Password:

     Welcome to VAX/VMS Version 5.0 on node TRNTO

                    .
                    .
                    .
$ LOGOUT
SMITH logged out at 30-DEC-1988 12:31:55:49

%REM-S-END, control returned to node _BOSTON::

$
```

When you are logged in at a remote node, you can use the SET HOST command to establish communication with another node. After logging in to node TRNTO, you could use SET HOST again to log in to another node (for example, node DENVER).

You would again be prompted for a user name and password. If you then supply a valid user name and password for node DENVER, you are logged in.

Note that when you log out of node DENVER, control is returned to node TRNTO. You must log out of node TRNTO to return to your local node, BOSTON.

For more detailed information about the SET HOST command, see its description in the *VMS DCL Dictionary*.

8.3 Accessing Files on Remote Nodes

DECnet–VAX allows you to access files on remote nodes in your network as though these files were on your local node. You can use the DECnet–VAX facilities to access remote files by means of DCL commands and command procedures, and MACRO and higher-level language programs using VMS RMS or VMS system services directly.

8.3.1 Using DCL Commands and Command Procedures

You can use most DCL commands that perform file operations at a local node to perform these operations on remote nodes. For example, you can use the same DCL commands to obtain directory listings, manipulate files, and execute command procedures on remote nodes. Generally, you need only prefix a node name followed by two colons to the standard VMS file specification to access the remote file. For example:

```
$ TYPE TRNTO::WORK$:[DOE]LOGIN.COM
```

In this example, the TYPE command requests that the file LOGIN.COM in the directory WORK$:[DOE] at the remote node TRNTO be displayed on your local terminal.

Depending on the file protections that are established on the remote node, you may need to supply an access control string in the DCL command when performing the file operation. For example:

```
$ COPY TRNTO"DOE JOHN"::WORK$:[DOE]LOGIN.COM *.*
```

In this example, an access control string is supplied as part of the request for the COPY operation. For VMS operating systems, the access control string consists of a user name, followed by one or more spaces or tabs, and, optionally, one password and/or one account.

As with DCL, remote file accessing by higher-level languages is accomplished in a way that is transparent to the user. The only additional information that you need to specify is the name of the remote node containing the file or files that you want to access. Like DCL, higher-level language programs also employ the VMS RMS services to perform file access operations.

Command descriptions in the *VMS DCL Dictionary* include restrictions that apply to individual commands and command qualifiers used in network operations. Unless otherwise stated, you can assume that a particular DCL command is supported for network operations.

8.3.2 Using Higher-Level Language Programs

You can use various higher-level languages to write programs that access remote files using the standard I/O statements of these languages. Regardless of the programming language used, you access remote files exactly as you would access local files.

In the following example, assume you want to design a FORTRAN program to transfer files from a local node to a remote node. You can identify the source and destination files by defining the logical names SRC and DST, respectively. You can use these DCL commands by entering the following commands:

```
$ DEFINE SRC TRNTO::INVENTDISK$:[STOCKROOM.PAPER]INVENTORY.DAT
$ DEFINE DST BOSTON::ARCDISK$:[ARCHIVE]TRNTO_INVENTORY.DAT
```

After you make the logical name assignments, the FORTRAN program can open the files by way of those logical names. You can use the following FORTRAN open calls:

```
OPEN (UNIT=1,NAME='SRC',TYPE='OLD',ACCESS='SEQUENTIAL',
     FORM='FORMATTED')

OPEN (UNIT=2,NAME='DST',TYPE='NEW',ACCESS='SEQUENTIAL',
     FORM='FORMATTED')
```

This FORTRAN program fragment uses standard I/O statements to transfer records from one file to another. In this example, the access mode is sequential.

As shown in the next example, you can design a FORTRAN program to transfer a file from the local node to a line printer on the remote node. You can define logical names for the source and destination, as follows:

```
$ DEFINE SRC TRNTO::INVENTDISK$:[STOCKROOM.PAPER]INVENTORY.DAT
$ DEFINE DSTLPR BOSTON::LPA0:
```

After you make the logical name assignments, the FORTRAN program can open the file and access the line printer by way of those logical names, as follows:

```
OPEN (UNIT=1,NAME='SRC',TYPE='OLD',ACCESS='SEQUENTIAL',
     FORM='FORMATTED')

OPEN (UNIT=2,NAME='DSTLPR',TYPE='NEW',ACCESS='SEQUENTIAL',
     FORM='FORMATTED',CARRIAGECONTROL='LIST',
      RECORDTYPE='VARIABLE')
```

This FORTRAN program fragment uses the standard I/O statements to transfer records from the source file to the destination line printer. The access mode of the file is sequential.

Examples of complete higher-level language programs designed to access remote files are included in the appropriate sections of the programming manuals for each VAX language.

8.3.3 Using RMS Services from MACRO Programs

The VMS operating system provides a programming interface for remote file access using higher-level languages, including VAX MACRO. The MACRO programs can use VMS Record Management Services (RMS) calls or VMS system service calls. This section describes how you can use RMS to access remote files. The VMS system services, which you can also use for remote file access, are described more completely in Section 8.5.4.

For remote file processing, RMS integrates the network software necessary to translate standard RMS calls, which provides a transparent user interface to the network.

Using the RMS facilities, you can perform remote file-handling operations on entire files or access individual records, through programmed RMS service calls in a VAX MACRO application. All you need to do is supply the name of the remote node in your file specification.

As in the previous FORTRAN examples, you can use DCL commands to make logical name assignments to the source and destination files that you want to manipulate, for example:

```
$ DEFINE SRC TRNTO::INVENTDISK$:[STOCKROOM.PAPER]INVENTORY.DAT
$ DEFINE DST BOSTON::ARCDISK$:[ARCHIVE]TRNTO_INVENTORY.DAT
```

Before you can open either the source (SRC) or destination (DST) file with the RMS $OPEN statement, however, you must allocate the appropriate file access blocks (FABs) and record access blocks (RABs) in your program. To do this, you can use the following RMS structures:

```
                .
                .
                .
SRC_FAB:
        $FAB    FAC=GET,-
                FOP=SQO,-
                FNM=SRC

SRC_FAB:
        $RAB    FAB=SRC_FAB,-
                RAC=SEQ,-
                .
                .
                .
```

These statements define the source file FAB and RAB control blocks. You must also define the destination file FAB and RAB control blocks, as follows:

```
                .
                .
                .
DST_FAB:
        $FAB    FAC=PUT,-
                FOP=SQO,-
                FNM=DST,-
                ORG=SEQ,-
                RFM=VAR,-
                RAT=CR
DST_RAB:
        $RAB    FAB=DST_FAB,-
                RAC=SEQ,-
                    .
                    .
                    .
```

After defining the source and destination FABs and RABs, you can open the files for remote file processing. Note that, if your program accesses files sequentially, you can specify the sequential-only (SQO) option of the file options (FOP) field of the FAB. Specifying FOP=SQO enables RMS and the remote File Access Listener (FAL) to enter into file-transfer mode. In file-transfer mode there is no wait for message acknowledgment and, consequently, there is a significant increase in file-transfer performance.

The *Guide to VMS File Applications* contains examples of complete MACRO programs using RMS to access remote files. Examples in this document also illustrate the network-specific features provided by VMS RMS.

The *VMS Record Management Services Manual* and the *VMS System Services Reference Manual* describe the RMS fields and options that you must specify for DECnet–VAX applications. These manuals also describe restrictions that apply to using RMS over the network. See Chapter 9 for a list of restrictions on VMS operations involving other systems in a heterogeneous network.

Note that DECnet–VAX does not support the use of RMS for operations on a remote magnetic tape volume.

8.4 Performing Task-to-Task Operations

Task-to-task communication is a feature common to all DECnet implementations. It allows two programs or tasks running under the same or different operating systems to communicate with each other regardless of the programming languages used. For example, a FORTRAN task running on the VMS operating system at node BOSTON could exchange messages with a MACRO task running on the RSX–11M operating system at node DALLAS. Although these programs use different programming languages and run under different operating systems, the DECnet software translates system-dependent language calls into a common set of network protocol messages.

8.4.1 Transparent and Nontransparent Task-to-Task Communication

DECnet–VAX supports both transparent and nontransparent task-to-task communication. Transparent communication provides the means for a DCL command procedure or a user program (written in either VAX MACRO or in a higher-level language) to communicate with other command procedures or user programs over the network, with no knowledge of the DECnet–VAX software. Nontransparent communication allows the programmer to use system service options to perform network-specific functions.

There are important differences between these two forms of communication. Transparent communication is a form of device-independent I/O in VMS in which you move data with little concern for the way the operation is accomplished. Likewise, transparent communication allows you to move data across the network without necessarily knowing that you are using DECnet software. Nontransparent communication, on the other hand, is a form of device-dependent I/O, in that you are interested in specific characteristics of the device that you want to access. A nontransparent task, in turn, can use network-specific features to monitor the communication process.

Note: While it is possible for a single task to create and maintain both transparent and nontransparent connections, each connection should be processed separately. That is, transparent-specific RMS and system services apply to transparent links, and nontransparent-specific system services apply to nontransparent links.

8.4.1.1 Transparent Communication

Transparent communication provides the basic functions necessary for a task to communicate with another task over the network. These functions include the initiation and completion of a logical link connection, the orderly exchange of messages between both tasks, and the controlled termination of the communication process. To perform these functions, you can write your cooperating tasks in any of the higher-level languages supported over the network, in VAX MACRO (using RMS service calls or system service calls), or by using DCL commands.

One way to view transparent communication is to look at the programming required to develop such an application. Transparent access provides the functions necessary to communicate over the network using standard I/O operations. When accessing the network transparently, you may use standard I/O statements of the higher-level language or straightforward RMS or system service calls to access a sequential record-oriented device. System service calls are described in Section 8.5.

8.4.1.2 Nontransparent Communication

Nontransparent communication provides the same functions as transparent communication plus additional system service and I/O features supported by DECnet–VAX. In particular, a nontransparent task can create and use a VMS mailbox to receive information that is not available to a transparent task with transparent communication. You can make use of network-specific features such as optional user data on connects and disconnects, and *interrupt messages*. Also, nontransparent tasks can receive and process multiple inbound connection requests. (See the description in Section 8.6.1.5.)

Note that on a VAXcluster node, nontransparent tasks that can receive multiple inbound connection requests should not use the cluster alias node address for outgoing connections, and should not be enabled to receive incoming connections directed to the cluster alias node. Incoming links directed to a cluster alias node address can be assigned to any of the nodes in the cluster that accept that alias node address, without knowledge of the nodes on which a declared task may be running (see Section 2.6.2).

In general, nontransparent tasks can use a mailbox to receive information about particular network operations. There are four types of mailbox messages:

- Messages that result from the use of certain system service calls (including optional user data carried on logical link creation or termination)
- Interrupt messages
- Logical link status messages
- Network system messages

Nontransparent functions that indirectly cause mailbox messages to be placed in the receiver's mailbox include calls for initiating, completing and terminating logical links. Figure 8–1 illustrates how nontransparent tasks use mailboxes.

Table 8–3 provides a list of mailbox messages and their meanings.

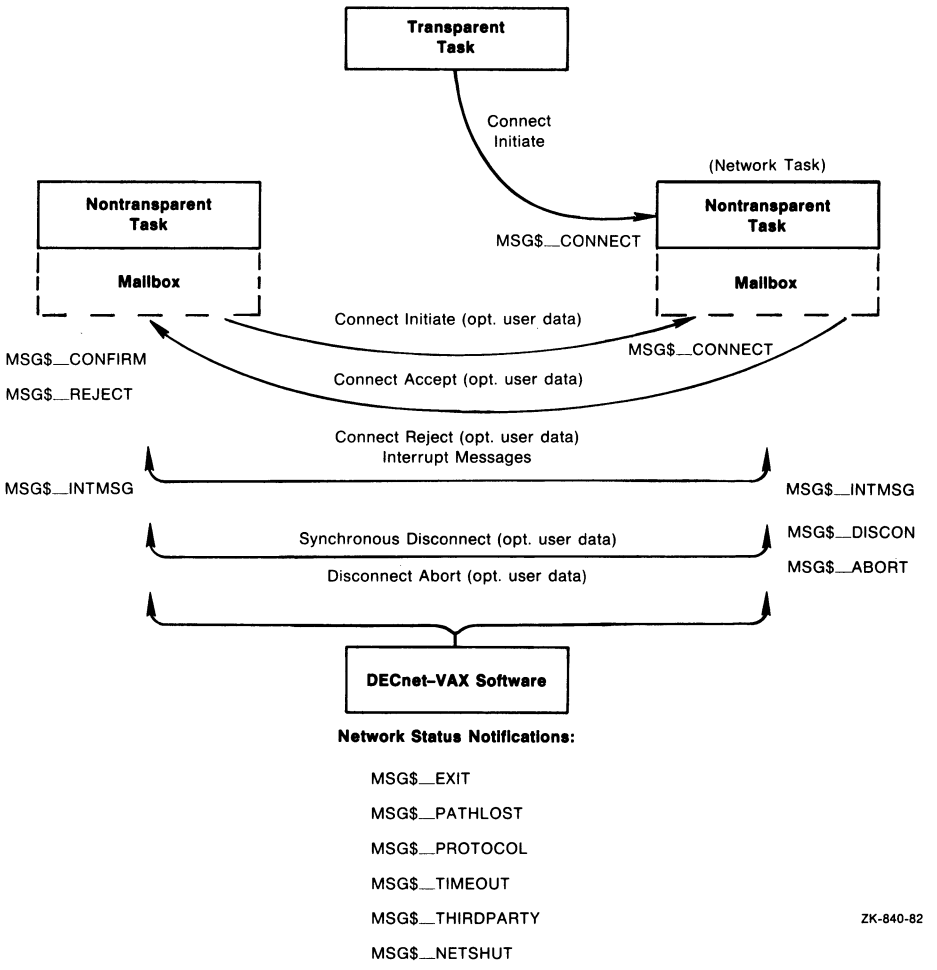
A nontransparent task can receive **network status notifications** in the mailbox. These notifications apply to physical and logical link conditions over the network. For example, DECnet–VAX software can notify a nontransparent task of the following conditions:

- Third-party disconnections
- Network software- and hardware-related problems
- Processes exiting before I/O completion
- Connection request timeouts

8.4.2 Task Specification Strings in Task-to-Task Applications

Whether you are performing a transparent or nontransparent task-to-task operation, you must use a **task specification string** to identify the remote task with which you want to communicate. A task specification string is a quoted string that identifies the target task to which you attempt a logical link connection.

Figure 8–1 Mailbox Messages



To establish a logical link connection with a target task addressed as object type 0, use either of the following forms of task specification string:

- *"TASK=taskname"*
- *"0=taskname"*

where:

taskname Can be from 1 to 12 characters.

Note that "0" and "TASK" are equivalent. (If the remote node is not a VMS system, the maximum length of the taskname may be different.)

If the remote node is a VMS operating system, the *taskname* usually represents the file name of a command procedure to be executed at the remote node. The *taskname* may also represent a specific image to be run. The command procedure invoked at the remote node can complete the logical link itself (using a DCL OPEN command), or it can include a DCL RUN command to execute a program that completes the logical link.

The examples that follow illustrate two uses of the task specification string. The first example identifies the task TEST2 by using the "TASK=" form for specifying target tasks. The second example is the same as the first, except that access control information is provided and the alternative "0=" form for specifying a task is used.

```
BOSTON::"TASK=TEST2"

BOSTON"SMITH JOHN"::"0=TEST2"
```

In this example, TEST2 refers to SYS$LOGIN:TEST2.COM for the default DECnet account at the remote VMS node. Note that only the file name component of the command file specification is used in the task name string in this example. When naming the target task, you can specify a more complete file specification. For example, you can include a device name or a file type.

8.4.3 Functions Required for Performing Task-to-Task Operations

Several functions are necessary for performing a task-to-task operation. The number of functions, of course, depends on whether you intend to access the network transparently or nontransparently.

Even a transparent task-to-task application requires a minimum number of operations to initiate and complete a logical link connection, to exchange messages, and to terminate the logical link. These operations are actually a subset of a larger group of functions defined for nontransparent communication. The entire set of functions is as follows:

- **Initiating a logical link connection**
  - Requesting a logical link to a remote task[1]
  - Declaring a network name and processing multiple connection requests
- **Completing a logical link connection**
  - Rejecting a logical link connection request
  - Accepting a logical link connection request[1]
- **Exchanging messages**
  - Sending and receiving data messages[1]
  - Sending and receiving interrupt messages
- **Terminating a logical link**
  - Synchronously disconnecting the logical link
  - Aborting the logical link[1]

Nontransparent tasks can use any or all of these functions to extend the basic capabilities offered under transparent communication.

[1] This operation represents the minimum subset for transparent task-to-task communication.

8.4.3.1 Initiating a Logical Link Connection

Whether you access the network transparently or nontransparently, you must establish a communication link to the remote node on which the target task runs before any message exchange can take place. You establish the link by issuing a source task call that requests a logical link connection. (The **source task** is the task that initiates a logical link connection request; the **target task** is the task with which you want to communicate.)

The interaction between the source task and the target task that takes place before the logical link is established is called a **handshaking sequence**. Upon receiving a call that requests a logical link connection, the local DECnet–VAX node initiates a handshaking sequence with the target task. The following information is supplied in a connection request:

- An I/O channel. The I/O channel (more commonly referred to as the channel) serves as the path over which messages are sent and received by the source task.
- The identification of the target node. Every node in a network has an identifier that distinguishes it from all other nodes in the network. Transparent communication uses a task specification string to indicate the name of the target node. Nontransparent communication requires a user-generated data structure called the **network connect block (NCB)**, which also includes a task specification string.
- An object type descriptor.
- Access control information (optional).
- Optional user data. Nontransparent tasks have the option of sending up to 16 bytes of data to the target task (see the following information about NCBs).

You should be aware that after you issue a call that uses either a task specification string or an NCB, you access the network and, by definition, the DECnet–VAX software.

8.4.3.2 Completing the Logical Link Connection

As part of the handshaking sequence, the target task completes the logical link connection in two steps. First, the DECnet software at the remote node processes the inbound logical link connection request. Second, the target task either accepts or rejects the link. These steps are performed differently, depending on whether the target task uses transparent or nontransparent I/O.

When a logical link request is received, a procedure called NETSERVER.COM is executed, which in turn invokes the image NETSERVER.EXE. This program works in conjunction with the network ACP (NETACP) and uses DCL to invoke the image or command procedure defined for the requested object. (For example, the specified task is invoked for object 0 and FAL is invoked for object 17.)

When the logical link is terminated, the "object" program (for example, FAL) also terminates. However, the process is not deleted. Instead, control is returned to NETSERVER.EXE, which communicates with NETACP to inquire for another incoming logical link request. This inquiry process continues until NETSERVER encounters a timeout condition (the default is 5 minutes).

The system manager can specify the time that NETSERVER waits for another logical link request. The logical name NETSERVER$TIMEOUT, when defined, determines the amount of time NETSERVER waits before reaching the timeout condition. Note that the equivalence name must be in the standard VMS delta time format, for example, 0:10:0, representing 10 minutes.

You may define a number of NETSERVER processes that never time out. This is useful on systems that are the target of significant amounts of network activity, such as mail or public file access. Two benefits may be gained: improved response time for the user initiating the network access, because there is no waiting for a new process to be created, and reduced overhead on the target system by virtue of fewer process creations.

To allow for permanent servers, define the logical name NETSERVER$SERVERS_*username* in the login procedure for the account receiving the network connects. The translation of the logical name should be the number of permanent servers you want. For example, to define two permanent servers for the default DECnet account (user name DECNET), enter the following command:

```
$ DEFINE NETSERVER$SERVERS_DECNET 2
```

You should put this command in the login command procedure of the default DECnet account. You could also define it as a system logical name in the site-dependent system startup command procedure. The account must have write access to its SYS$LOGIN directory. Note that you gain very little by defining only one permanent server, because a number of functions such as wildcard file copy require multiple logical links, each of which requires its own server.

If you use this mechanism, you should understand the interaction between proxy access and NETSERVER processes. The proxy database is read by LOGINOUT.EXE, after a process has been created. For this reason, any incoming connection that may have a proxy account on the local system will not be given to an existing NETSERVER process that was created for a different user. Permanent servers, in general, can be used only by logical links that are not using proxy access.

In the following discussion, the remote node is assumed to be a VMS operating system. If the remote node on which your target task runs is not a VMS operating system, you should refer to the DECnet documentation for that system.

Completing the Connection Transparently

If the target task is transparent, the DECnet software at the remote node checks the access control information supplied in the connection request call.

Before you access the remote node, the system manager must have created the appropriate account in the UAF (refer to the information on access control). In addition, the command procedure file (*taskname*.COM) starting the remote task must exist in the default directory associated with the account identified by the access control information. For a description of the command procedure *taskname*.COM, see Section 8.7.1, which contains examples of command procedures designed for task-to-task communication.

Command procedures for objects existing in the OBJECT database (which is created using NCP commands) are located in the SYS$SYSTEM directory. The DIGITAL-supplied FAL.COM procedure is an example of such a command procedure. (Note that the object command procedure is bypassed if the object definition specifies an EXE file.)

Completing the Connection Nontransparently

If the target task is nontransparent, then one of several things may occur. If the task has not declared itself a **network task** (and is therefore eligible to accept only one connection request at a time), then the DECnet software at the remote node performs the access checking procedure. After it starts, the target task retrieves the connection information by translating the logical name SYS$NET using the $TRNLNM system service call (see Section 8.6).

If the target task declares itself as an active network task, then DECnet–VAX software places all connection requests addressed to the task in the mailbox associated with the channel being used. The first message in the mailbox is the NCB from the original connection request that started the task. This message appears in the mailbox after channel assignment and name declaration occur. After the task declares a network name or number, subsequent inbound connection requests are not checked by the remote node to verify access control. (Note that if the task is started without being part of a DECnet operation, access control is never checked.) Section 8.6 describes in more detail the nontransparent process of completing the logical link connection.

After examining the incoming connection request, the target task either accepts or rejects the request, and optionally can send 1 to 16 bytes of data back to the source task at the same time that it responds to the logical link connection request. Furthermore, a library routine, LIB$ASN_WTH_MBX, which assigns a channel and associates a unique mailbox, can be used when accepting the connection.

8.4.3.3 Exchanging Messages

When you access the network transparently or nontransparently, DECnet–VAX sends data messages over a logical link in response to a set of send and receive calls issued by the source and target tasks. For higher-level language tasks, use standard read and write statements to send and receive data messages. (In Example 8–2, the two FORTRAN tasks use READ and WRITE statements to exchange information. The equivalent RMS service calls are $GET and $PUT.)

After DECnet–VAX creates a logical link, the two tasks are ready to exchange messages. This exchange can take place only if the two tasks cooperate in the transmission process. In other words, for each message sent by a task, the receiving task must issue a corresponding call to receive the message. Also, you must decide which task will disconnect the link. In addition, if the tasks are nontransparent, they must agree on whether or not the optional data will be passed. In the context of an established logical link, the task sending a message is the transmitter and the task receiving it is the receiver. Because logical links are inherently full duplex, each task may be a transmitter and a receiver simultaneously.

DECnet–VAX distinguishes between two types of message: data messages and mailbox messages. Data messages are the normal mode of information exchange for both transparent and nontransparent communication. Mailbox messages such as interrupt messages, messages resulting from some DECnet

operation (including optional user data), and network status notifications, can be used only in nontransparent communication.

Nontransparent communication frequently involves using a mailbox to obtain network-specific information. A task may receive three types of message in its mailbox:

- Messages that DECnet generates when the task initiates certain network operations. A VMS task issues system service calls to initiate these operations. For example:
  - When one task requests a logical link connection, a notification message (and optional user data) may be placed in the mailbox of the target task.
  - When a target task accepts or rejects the logical link connection request, a notification message (and optional user data) is placed in the mailbox of the source task.
  - When one task synchronously disconnects or aborts a logical link, a notification message (and optional user data) is placed in the mailbox of the task from which it is disconnecting.
- Network status notification messages that inform a task of some unusual network occurrence (such as a third-party disconnect).
- Interrupt messages sent by the other task.

8.4.3.4 Terminating a Logical Link Connection

The termination of a logical link signals the end of the communication between tasks.

In transparent communication using higher-level language statements, RMS service calls, or system service calls, either task can break the link. To terminate the link properly, the receiver, and not the transmitter, of the final message should issue the $CLOSE service to break the link. The link termination process is complete when the other task issues a link termination request. In transparent communication using system service calls, the $DASSGN system service call causes the link to be terminated.

Issuing the $CANCEL service call followed by the $DASSGN service call causes all pending operations to abort, then closes the link and deassigns the channel.

In nontransparent communication using system service calls, you can terminate I/O operations over a channel in one of three ways:

- **Synchronous Disconnect ($QIO)**—Specifies that all messages sent by the local task are required to be received and acknowledged by the remote End Communication Layer (ECL) before the logical link is disconnected. You should use this type of disconnect when the user of the logical link's services wants to ensure that the transmission of messages has completed before taking down the logical link. Note, however, that this service cannot guarantee the delivery of the received data to the remote task.
- **Disconnect Abort ($QIO)**—Specifies that all messages sent by the local task are not required to be received or acknowledged by the remote ECL before the logical link is disconnected. You should use this type of disconnect when the local task wants to reset the logical link to a known state. To ensure that the transmitted messages have been received and acknowledged by the remote ECL, the local task may issue the system service $CANCEL on the channel before issuing the disconnect abort.

Note, however, that these services cannot guarantee the delivery of the received data to the remote task.

- **Deassign Channel and Terminate Link ($DASSGN)**—Specifies that all messages sent by the local task are not required to be received or acknowledged by the remote ECL before the logical link is disconnected. You should use this type of disconnect when the local task wants to break a logical link and deassign the channel to the network immediately.

Note that after either a synchronous disconnect or a disconnect abort of a nontransparent link, you can issue a new connection request because you did not deassign the I/O channel but merely deaccessed the link. For further information about these system service calls, see Section 8.6.

When a connection to a nontransparent task terminates the connection, a notification message indicating that the link is disconnected is placed in the mailbox of the affected task. A nontransparent task can send up to 16 bytes of optional user data, with the disconnect request. This optional user data is placed in the mailbox of the nontransparent task on the receiving end of the disconnect message.

Disconnect operations cannot guarantee to both partners that communication is complete. Therefore, DIGITAL recommends that the communicating tasks agree on a protocol for terminating communication. In general, the receiver, not the transmitter, of the final message should disconnect the logical link.

Transparent communication allows you to create a logical link between tasks, send and receive data messages, and terminate the logical link at the end of the message dialog. The discussion covers general concepts implicit in DECnet–VAX task-to-task communication and assumes familiarity with the QIO-related material in the *VMS System Services Reference Manual*. The use of higher-level language statements and RMS service calls in transparent task-to-task communication is described in Section 8.5.

8.5 Performing Transparent Task-to-Task Operations

This section describes the system service calls and functions that you can use to perform transparent task-to-task communication over the network. You can perform these operations using any of the following methods:

- DCL commands and command procedures
- Higher-level language programs using appropriate language I/O statements
- MACRO or higher-level language programs using VMS RMS calls or VMS system service calls

See Section 8.7 for examples of transparent task-to-task operations.

8.5.1 Using DCL Commands and Command Procedures

To perform transparent task-to-task operations, you can use DCL commands to construct and execute command procedures.

For example, to display information about another system, you can design a command procedure that can be invoked as a remote task. Assume that a procedure called SHOWBQ.COM is designed to return status information about jobs entered in batch queues on the system where it executes. Assume also that SHOWBQ.COM resides on node TRNTO. You can use SHOWBQ.COM for task-to-task communication by entering a task specification string in a TYPE command. For example:

```
$ TYPE TRNTO"BROWN JUNE"::"TASK=SHOWBQ"
```

See Section 8.7.1 for an example of a command procedure used for task-to-task communication. For additional information concerning the design, construction, and execution of command procedures, see the *Guide to Using VMS Command Procedures*.

8.5.2 Using Higher-Level Language Programs

This section contains examples of higher-level language calls that you can use for transparent task-to-task communication. Each higher-level language call contains a task specification string as part of its statement.

Higher-level language tasks can use standard file opening statements to request a logical link connection to a remote task. The following examples show how to specify a target task, TEST4, running on node TRNTO, in various languages supported on the VMS operating system.

FORTRAN
```
OPEN (UNIT=7,NAME='TRNTO::"TASK=TEST4" ',TYPE='NEW')
```

BASIC
```
OPEN 'TRNTO::"TASK=TEST4"' AS FILE #7
```

PL/I
```
OPEN FILE(OUTPUT) TITLE ('TRNTO::"TASK=TEST4"');
```

PASCAL
```
OPEN (PARTNER,'TRNTO::"TASK=TEST4"',NEW);
```

COBOL
```
SELECT PARTNER ASSIGN TO "TRNTO::""TASK=TEST4""". OPEN OUTPUT PARTNER.
```

C
```
F1 = OPEN ("TRNTO::\"TASK=TEST4\ "",2);
```

To complete the logical link, the target task performs a file opening operation using the logical name SYS$NET to establish a communications path back to the source task. The following examples show how to specify SYS$NET from higher-level language calls.

FORTRAN
```
OPEN (UNIT=2,NAME='SYS$NET',TYPE='OLD')
```

BASIC
```
OPEN "SYS$NET" AS FILE #2
```

PL/I
```
OPEN FILE(INPUT) TITLE ('SYS$NET');
```

PASCAL
```
OPEN (PARTNER,'SYS$NET',OLD);
```

COBOL
```
SELECT PARTNER ASSIGN TO "SYS$NET". OPEN INPUT PARTNER.
```

C
```
F2 = OPEN ("SYS$NET",2);
```

Section 8.7.2 provides an example of a FORTRAN program designed for transparent task-to-task communication.

8.5.3 Using RMS Service Calls in MACRO Programs

You can write a MACRO program or a higher-level language program to perform transparent task-to-task communications, using RMS service calls. This section describes how to use RMS service calls in a MACRO program.

Note that the RMS $OPEN statement is equivalent to the higher-level language statements described in Section 8.5.2.

After you define the appropriate FAB and RAB control blocks, you can use the $OPEN statement to specify the target task, TEST4, running on node TRNTO. You can initiate the link by specifying the following call, in your MACRO program:

```
TARGET:
        $FAB    FAC=<GET,PUT>,-
                ORG=SEQ,-
                FNM=<NODE::"TASK=TEST4">
        $OPEN   FAB=TARGET
```

To complete the logical link, the target task performs a file-opening operation using the logical name SYS$NET to establish a communications path back to the source task. For example:

```
REQUESTER:
        $FAB    FAC=<GET,PUT>,-
                ORG=SEQ,-
                FNM=<SYS$NET>
        $OPEN   FAB=REQUESTER
```

As in the case of the target task, the appropriate FABs and RABs must already be declared, if the RMS OPEN call is to succeed. On inbound connections, DECnet–VAX automatically makes the logical name assignment to SYS$NET.

8.5.4 Using System Service Calls in MACRO Programs

You can write MACRO programs or higher-level language programs to perform transparent task-to-task communications, using system service calls. This section focuses on MACRO programs using system service calls for performing these operations.

Table 8–1 summarizes these calls and their network-related functions. Section 8.5.5 presents the format of these calls in more detail.

Table 8–1 System Service Calls for Transparent Communication

Call	Function
$ASSIGN	Request a logical link connection
$DASSGN	Terminate a logical link
$QIO (IO$_READVBLK)	Receive a message
$QIO (IO$_READVBLK!IO$M_MULTIPLE)	Receive a message in multiple receive requests
$QIO (IO$_WRITEVBLK)	Send a message
$QIO (IO$_WRITEVBLK!IO$M_MULTIPLE)	Send a message in multiple write requests

These calls allow you to perform task-to-task communication in much the same way as you would perform normal I/O operations. Use the $ASSIGN call to assign a logical link I/O channel to a device, which in this case is a task that behaves like a full-duplex record-oriented device. You can perform read and write operations with this task either synchronously or asynchronously. To exchange messages, use the Queue I/O (QIO) requests supported by DECnet–VAX. When all communication completes, use the $DASSGN system service call to deassign the channel and thereby disconnect the logical link.

8.5.4.1 Requesting a Logical Link

To request a logical link and assign an I/O channel, use the $ASSIGN system service. When you issue this call, you must include a task specifier for the remote node on which the *cooperating task* runs. The task specifier identifies the remote node and the target task to which you want to establish a logical link.

For example, for the network model described in Chapter 1, you could establish a logical link to target task TEST2 on node TRNTO to perform task-to-task communication. To create this link, code the following VAX MACRO statements in your source program.

```
TARGET:     .ASCID  /TRNTO::"TASK=TEST2"/
NETCHAN:    .BLKW   1       ; Channel number returned here
                .
                .
                .
            $ASSIGN_S       DEVNAM=TARGET,CHAN=NETCHAN
```

For debugging or for symmetry, you can develop and run the target task on the local node. Use the local node name (or node number 0) plus two colons to connect to the local node. This practice applies to DCL, higher-level languages and RMS, as well as system services.

After you establish a logical link, you refer to the assigned channel in any succeeding call in the MACRO program, either to send or receive messages, or to deassign the channel and terminate the logical link.

Until the connection operation completes, the process is in local event flag wait (LEF) state in kernel mode. Therefore, pressing CTRL/Y does not return the process to DCL status. The maximum amount of time that the process will wait in this state is specified by the OUTGOING TIMER parameter of the NCP command SET EXECUTOR. If this timer cannot be set to an acceptable

value, tasks that accept commands from the terminal should use $QIO (IO$_ACCESS) instead of the transparent $ASSIGN call to initiate logical links.

8.5.4.2 Completing the Logical Link Connection

The target task completes the logical link by specifying the logical name SYS$NET as the *devnam* argument for the $ASSIGN system service. For example:

```
LOGNAM:     .ASCID  /SYS$NET/
NETCHAN:    .BLKW   1       ; Channel number returned here
                .
                .
                .
            $ASSIGN_S       DEVNAM=LOGNAM,CHAN=NETCHAN
```

Issue this call in the target task to complete the logical link connection. The target task also specifies a channel to be used in subsequent system service calls.

The remote node is assumed to be a VMS operating system. If the remote node on which the target task runs is other than VMS, you should refer to the related DECnet documentation.

8.5.4.3 Exchanging Messages

After DECnet–VAX software establishes a logical link with the target task, either task can then send or receive messages. However, they must cooperate with each other: for each message sent with the $QIO (IO$_WRITEVBLK), the other task must issue a corresponding $QIO (IO$_READVBLK) to receive the message.

On logical links, DECnet–VAX supports sending and receiving data messages that are larger than the maximum size allowed by the $QIO system service. You do this by allowing write and read requests to be fragmented across multiple $QIO requests. To fragment writes and reads, you must include the modifier IO$M_MULTIPLE on the write or read $QIO call.

When you supply the modifier on a write message request $QIO (IO$_WRITEVBLK!IO$M_MULTIPLE), it indicates that more data will be supplied for this message. To indicate the last fragment of the message being sent, you should issue the write request without a modifier $QIO (use the QIO called IO$_WRITEVBLK).

When you supply the modifier on a read message $QIO (IO$_READVBLK!IO$M_MULTIPLE), if the received data message contains more than enough data to fill the buffer supplied with the read request, then SS$_BUFFEROVF is returned. This is not an error status. The next read posted receives the next fragment of the data message. If the received message fits into the buffer posted, then SS$_NORMAL is returned. Tasks that require fragmentation should always supply the IO$M_MULTIPLE on read requests.

If you do not use the read multiple request to receive a data message, then you must ensure that the tasks allocate enough buffer space for receiving the messages. If the tasks do not, a SS$_DATAOVERUN error occurs. You must also ensure that the end of the dialog can be determined.

One of the two tasks must disconnect the logical link. To terminate a logical link properly, the receiver, and not the transmitter, of the final message should break the link.

DECnet–VAX does not provide an automatic timeout of read or write requests. If the task needs to stop a read or write request on a logical link, then it must do so by disconnecting or aborting the logical link.

8.5.4.4 Terminating the Logical Link

Use the $DASSGN system service call to deassign the channel and break off the logical link with the cooperating task. This call terminates all pending calls for sending and receiving messages, aborts the link immediately, and frees the channel associated with that logical link.

8.5.4.5 Status and Error Reporting

When a system service completes execution, a status value is returned (does not apply to the $EXIT service). The $ASSIGN, $DASSGN, and $QIO system services place the return status information in register 0 (R0). For the $QIO system service, a successful return status indicates only that the request was queued successfully. All I/O completion status information is placed in the I/O status block (IOSB). For example, a $QIO system service read request to a task might be successful (status return is SS$_NORMAL) yet fail because the link was disconnected. (I/O status return is SS$_LINKABORT.) The return status codes shown in the following sections may be returned both in R0 and in the IOSB.

When DECnet–VAX returns the status SS$_NORMAL in the I/O status block on a write request, it means that the write was queued for transmission on the logical link. It does not mean that the write request has been received or acknowledged by the remote task. The logical link services of DECnet–VAX provide the guaranteed delivery of transmitted messages to the remote node. If a message cannot be delivered, the user is notified by the disconnection of the logical link. The DECnet–VAX services cannot guarantee the delivery of data received on the remote node to the remote task. It is the responsibility of cooperating tasks to agree on a protocol to ensure that data transmitted by the local task is received by the remote task.

The *VMS System Services Reference Manual* and the *Guide to VMS Programming Resources* both provide more information about $QIO system services.

8.5.5 Summary of System Service Calls for Transparent Operations

The following sections describe the VMS system services you can use for transparent task-to-task communication. Each description covers the use of the call, its format, the arguments associated with the call, and the return status information. The *VMS System Messages and Recovery Procedures Reference Volume* lists the entire set of network system service error messages.

8.5.5.1 $ASSIGN

The $ASSIGN system service assigns a channel to refer to the logical link. You can then use the channel returned in the *chan* argument in any succeeding call to send or receive a message, or to deassign the channel and thereby terminate the logical link.

Format

```
$ASSIGN devnam ,chan ,[acmode]
```

Arguments

devnam — Address of a quadword descriptor of a character string that identifies the remote task. The string contains either of the following:

- A task specification string if the call is by the source task. Both the string and its descriptor must be in read/write storage.
- The SYS$NET logical name if the call is by the target task.

chan — Address of a word that is to receive the assigned channel number. You use this channel number to send a message to a remote task, receive a message from a remote task, or to abort the logical link.

acmode — Access mode to be associated with this channel. The most privileged access mode used is the access mode of the caller. You can perform I/O operations on the channel only from equal or more privileged access modes.

Return Status

SS$_CONNECFAIL	The connection to a network object timed out or failed.
SS$_DEVOFFLINE	The physical link is shutting down.
SS$_FILALRACC	A logical link already exists on the channel.
SS$_INSFMEM	There is not enough system dynamic memory to complete the request.
SS$_INVLOGIN	The access control information was found to be invalid at the remote node.
SS$_IVDEVNAM	The task specifier has an invalid format or content.
SS$_LINKEXIT	The network partner task was started, but exited before confirming the logical link (that is, $ASSIGN to SYS$NET).
SS$_NOLINKS	No logical links are available. The maximum number of logical links as set for the NCP executor MAXIMUM LINKS parameter was exceeded.
SS$_NOPRIV	The issuing task does not have the required privilege to perform network operations or to confirm the specified logical link.
SS$_NOSUCHNODE	The specified node is unknown.
SS$_NOSUCHOBJ	The network object number is unknown at the remote node; or for a TASK= connect, the named DCL command procedure file cannot be found at the remote node.

SS$_NOSUCHUSER	The remote node could not recognize the login information supplied with the connection request.
SS$_PROTOCOL	A network protocol error occurred, most likely because of a network software error.
SS$_REJECT	The network object rejected the connection.
SS$_REMOTE	The service completed successfully. (A logical link was established with the target task.)
SS$_REMRSRC	The link could not be established because system resources at the remote node were insufficient.
SS$_SHUT	The local or remote node is no longer accepting connections.
SS$_THIRDPARTY	The logical link connection was terminated by a third party (for example, the system manager).
SS$_TOOMUCHDATA	The task specified too much optional or interrupt data.
SS$_UNREACHABLE	The remote node is currently unreachable.

8.5.5.2 $QIO (Sending a Message to a Target Task)

The $QIO system service with a function code of IO$_WRITEVBLK or IO$_WRITEVBLK!IO$M_MULTIPLE sends a message to a target task. The $QIO call initiates an output operation by queuing a request to the channel associated with the logical link. Alternatively, you could use the $QIOW system service to perform the same operation but also wait for I/O completion.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,p1 ,p2
$QIOW

Arguments

efn	Number of the event flag to be set at request completion.
chan	Word containing the channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_WRITEVBLK or IO$_WRITEVBLK!IO$M_MULTIPLE.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of an asynchronous system trap (AST) routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Buffer address.
p2	Buffer length in bytes.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_ABORT	The I/O request has been aborted by a $DASSGN or $CANCEL call.
SS$_CANCEL	The I/O on this channel has been canceled.
SS$_FILNOTACC	No logical link is associated with the channel.
SS$_INSFMEM	Enough memory to buffer the message could not be allocated.
SS$_LINKABORT	The network partner task aborted the logical link.
SS$_LINKDISCON	The network partner task disconnected the logical link.
SS$_LINKEXIT	The network partner task exited.
SS$_PATHLOST	The path to the network partner task node was lost.
SS$_PROTOCOL	A network protocol error occurred. This is most likely due to a network software error.
SS$_THIRDPARTY	The logical link connection was terminated by a third party (for example, the system manager).

8.5.5.3 $QIO (Receiving a Message from a Target Task)

The $QIO system service with a function code of IO$_READVBLK receives a message from a target task. The $QIO call initiates an input operation by queuing a request to the channel associated with the logical link. Alternatively, you could use the $QIOW system service to perform the same operation but also wait for I/O completion.

Format

```
$QIO   [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,p1 ,p2
$QIOW
```

Arguments

efn	Number of the event flag to be set at request completion.
chan	Word containing the channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_READVBLK or IO$_READVBLK!IO$M_MULTIPLE.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of an AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Buffer address.
p2	Buffer length in bytes.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_ABORT	The I/O request has been aborted by a $DASSGN or $CANCEL call.
SS$_CANCEL	The I/O on this channel has been canceled.
SS$_DATAOVERUN	More bytes were sent than could be received in the supplied buffer. This status will not be returned when IO$M_MULTIPLE is used on the read request.
SS$_FILNOTACC	No logical link is associated with the channel.
SS$_INSFMEM	Enough memory to buffer the message could not be allocated.
SS$_LINKABORT	The network partner task aborted the logical link.
SS$_LINKDISCON	The network partner task disconnected the logical link.
SS$_LINKEXIT	The network partner task exited.
SS$_PATHLOST	The path to the network partner task node was lost.
SS$_PROTOCOL	A network protocol error occurred. This is most likely due to a network software error.
SS$_THIRDPARTY	The logical link connection was terminated by a third party (for example, the system manager).
SS$_BUFFEROVF	Data could not fit in the buffer supplied. Supply another read request to receive the next fragment of received data message.

8.5.5.4 $DASSGN (Disconnecting a Logical Link)

The $DASSGN system service terminates all pending operations to send and receive data, disconnects the logical link immediately, and frees the channel associated with that link. Either task can terminate the logical link by calling $DASSGN.

Format

$DASSGN chan

Argument

chan — Word containing the channel number to the logical link you want disconnected. Use the same channel number returned in the $ASSIGN call.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_IVCHAN	The process specified an invalid channel.
SS$_NOPRIV	The specified channel was not assigned or was assigned from a more privileged access mode.

8.6 Performing Nontransparent Task-to-Task Operations

This section describes the system service calls and functions that you use for nontransparent task-to-task communication. In general, the principles of nontransparent task-to-task communication are similar to those of transparent communication.

If you want to perform nontransparent communication operations, you can write VAX MACRO programs using VMS system services designed specifically for DECnet–VAX. You can also write programs in one of the higher-level languages, provided the language supports the DECnet–VAX services. These DECnet–VAX services are described in detail throughout this section.

DECnet–VAX also provides additional services with extensions that allow you to use network-specific features for nontransparent network operations, such as the following:

- Creating and using mailboxes for receiving messages, including network status notifications
- Declaring a task as a network task, thus enabling it to process multiple inbound logical link connection requests
- Sending connection requests, optionally with user data
- Accepting or rejecting a connection request, optionally with user data
- Communicating between a transparent and a nontransparent task
- Sending or receiving an interrupt message
- Aborting or synchronously disconnecting a logical link, optionally with user data

The general concepts implicit in DECnet–VAX task-to-task communication are covered in Section 8.5. You should also be familiar with the material in the *VMS System Services Reference Manual* and the *VMS I/O User's Reference Volume*.

8.6.1 Using System Services for Nontransparent Operations

Nontransparent task-to-task communication over the network uses a set of system service calls available under the VMS operating system. Table 8–2 summarizes these calls and their network-related functions. The $QIO calls are distinguished by function code.

Table 8–2 System Service Calls for Nontransparent Communication

Call	Function
$ASSIGN	Assign an I/O channel
$CANCEL	Cancel I/O on a channel
$CREMBX	Create a mailbox
$DASSGN	Abort the logical link connection (deassigning an I/O channel)
$GETDVI	Get information on device or volume
$QIO (IO$_ACCESS)	Request a logical link connection
$QIO (IO$_ACCESS)	Accept a logical link connection request
$QIO (IO$_ACCESS!IO$M_ABORT)	Reject a logical link connection request
$QIO (IO$_ACPCONTROL)	Assign a network name to a task eligible to accept multiple inbound connection requests
$QIO (IO$_DEACCESS!IO$M_ABORT)	Abort the logical link connection
$QIO (IO$_DEACCESS!IO$M_SYNCH)	Synchronously disconnect a logical link
$QIO (IO$_READVBLK)	Receive a message
$QIO (IO$_READVBLK!IO$M_MULTIPLE)	Receive a message in multiple receive requests
$QIO (IO$_WRITEVBLK)	Send a message
$QIO (IO$_WRITEVBLK!IO$M_MULTIPLE)	Write a message in multiple write requests
$QIO (IO$_WRITEVBLK!IO$M_INTERRUPT)	Send an interrupt message
$TRNLNM	Translate logical names

8.6.1.1 Assigning a Channel to _NET: and Creating a Mailbox

To prepare for nontransparent task-to-task communication, you need to assign a channel just as you would for transparent communication. In addition, you can create a mailbox to take advantage of optional network protocol features.

You must assign a channel to the pseudodevice _NET:; use the $ASSIGN system service call for this purpose. This call normally contains a reference to a mailbox, thereby associating it with the channel assigned to _NET:. If you use a mailbox, you must create the mailbox before assigning a channel to _NET:. It is important to note that this use of the $ASSIGN system service differs from its use for transparent communication. Assigning a channel to _NET: does not transmit a logical link connection request to the remote node. Instead, the channel to _NET: provides a communication path to DECnet software. You must use a separate $QIO call (IO$_ACCESS function using the same channel) to request a logical link to the remote task. Refer to Section 8.6.2.1 for details about the $ASSIGN system service.

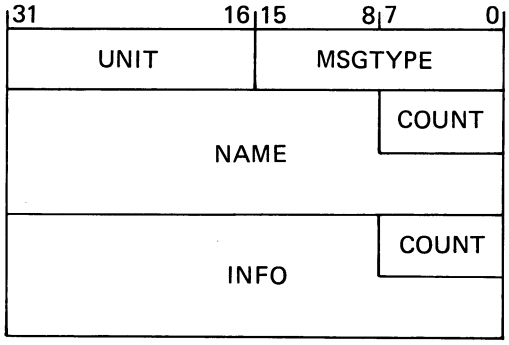
To take advantage of optional network protocol features, you can create a mailbox to receive messages on behalf of logical link operations. For example, the mailbox receives a message indicating whether the cooperating task accepted or rejected a connection request issued by the source task. Use the $CREMBX system service to create a mailbox for these purposes. In the event that your application does not need the information supplied in the mailbox, you need not create a mailbox.

For convenience, you can use the Run-Time Library routine LIB$ASN_WTH_MBX to create a temporary mailbox, assign a channel to it, and assign a channel to _NET:. This routine creates a unique mailbox on each call to the routine. Multiple copies of a task using this routine, in effect, use different mailboxes. If you were to create a mailbox with a logical name within the task, then all copies of that task would use the same mailbox and thereby interfere with each other's mailbox messages. For a complete description of this routine, see the *VMS Run-Time Library Routines Volume.*

8.6.1.2 Mailbox Message Format

The mailbox receives information specific to nontransparent communication with a remote task. Figure 8–2 illustrates the general format of the mailbox message.

Figure 8–2 Mailbox Message Format



ZK-841-82

Notes on Figure 8–2

MSGTYPE	Contains a code that identifies the message type.
UNIT	Contains the binary unit number of the device for which the message applies.
COUNT NAME	Contains a counted ASCII string that gives the name of the device for which the message applies. The $ASSIGN system service creates devices having names beginning with NET.
COUNT INFO	Contains a counted ASCII string of information, which depends on the message type.

All system mailbox messages contain, in the first word of the message, a constant that identifies the sender of the message. These constants have symbolic names (defined in the $MSGDEF macro) in the following format:

MSG$_sender

Table 8–3 summarizes the system mailbox messages that pertain to nontransparent task-to-task communication.

Table 8–3 System Mailbox Messages

Symbolic Name	Meaning
MSG$_TRMUNSOLIC	Unsolicited terminal data
MSG$_CRUNSOLIC	Unsolicited card reader data
MSG$_ABORT	Network partner aborted link
MSG$_CONFIRM	Network connect confirm
MSG$_CONNECT	Network inbound connect initiate
MSG$_DISCON	Network partner disconnected; hang-up
MSG$_EXIT	Network partner exited prematurely
MSG$_INTMSG	Network interrupt message; unsolicited data
MSG$_PATHLOST	Network path lost to partner
MSG$_PROTOCOL	Network protocol error
MSG$_REJECT	Network connect reject
MSG$_THIRDPARTY	Network third party disconnect
MSG$_TIMEOUT	Network connect timeout
MSG$_NETSHUT	Network shutting down

8.6.1.3 Requesting a Logical Link Connection

After you assign the I/O channel, you can request a logical link connection to the target task. Use the $QIO system service with a function code of IO$_ACCESS. You must identify the target task in the $QIO call. Use a network connect block (NCB) to specify the target task identification string. In addition, you can optionally send 1 to 16 bytes of data in the NCB. The format of the NCB is discussed in Section 8.6.1.4.

After the source task issues the connection request, it can issue a $QIO call with a function code of IO$_READVBLK to read its mailbox. Checking the contents of the mailbox is one way to determine whether the target task accepted or rejected the connection request. The mailbox can contain a variety of information, including either the MSG$_CONFIRM or MSG$_REJECT messages, and possibly optional data in the mailbox buffer.

If specified, the IOSB argument of the $QIO (IO$_ACCESS) call will also contain the result of the connection request operation. Section 8.6.2.2 provides a complete list of I/O status messages for this call.

Note that you must read the mailbox to inspect any optional data sent from the target task upon accepting or rejecting the connection request.

8.6.1.4 Using the Network Connect Block

The network connect block (NCB) is a user-generated data structure that contains the information necessary to request a logical link connection or to accept or reject a logical link connection request. The NCB must be in read/write storage.

The NCB identifies a specific task using a task specification string. This task specification string specifies either an object name or an object number. The following are valid task specification strings:

```
"TASK=TEST2
"TASK=157
"0=TEST2"
```

For an inbound call with an NCB, the task name portion of the task specification string is a process ID if the remote node is a VMS operating system; if not, then the task name portion is a system-specific string that identifies an executable unit (for example, job or task). The task specification string must be enclosed in quotation marks. Note that the final quotation mark of the task specification string follows the last item within the NCB. Section 8.4.2 provides additional information about task specification strings.

Example 8–1 shows an NCB you could use when issuing a connection request call. The significance of the information contained in the NCB block varies, depending on the type of call in which it is used. If the call is an outbound connection request with no optional data, items 2, 3, 4, and 5 of the block are not required. If the call is a connect accept operation and no optional data is sent, then items 4 and 5 are not required. Item 5 is meaningful only to the receiver of a connection request.

Example 8–1 Network Connect Block Format

```
1.  With optional data (outbound connect):
                                        ❶       ❷
     NCB:     .ASCII  ?TRNTO::"TASK=TEST2/?
              .WORD   0❸
     OPTDATA:
              .ASCIC /USERINFO/
              .BLKB 17-<.-OPTDATA>❹
              .ASCII /"/

                    ❺
2.  Without optional data (outbound connect):
                                        ❶
      NCB:      .ASCII   ?TRNTO::"TASK=TEST2"?
```

Item	Function
❶	A valid task specification string.
❷	The slash character (/).
❸	One word. This word must be 0 for a connection request operation. For a connect accept or reject operation, this word contains an internal DECnet link identifier.

❹ Up to 16 bytes of optional data sent as a counted string. This string is stored in a fixed-length field that is 17 bytes long. DECnet–VAX software ignores unused bytes.

❺ A destination descriptor. This descriptor indicates how the connection was issued and is meaningful only to the task or object to which the connection is made. This information is useful where one program serves many functions and needs to know how it was invoked. The maximum length for the destination descriptor is 19 bytes. The format is as follows:

a. If byte 0 contains 0, then byte 1 is the binary value of the object number.

b. If byte 0 contains 1, then byte 1 is the binary object number, and bytes 2 through 18 contain a counted task name.

c. If byte 0 contains 2, then byte 1 is the binary object number; bytes 2 through 5 contain a UIC, the first two bytes of which contain a binary group code and the second two bytes contain a binary user code; and bytes 6 through 18 contain a counted task name.

8.6.1.5 Completing the Establishment of a Logical Link

A nontransparent target task completes the logical link connection in one of several ways, depending upon whether the task can process multiple inbound connection requests or just a single request. Furthermore, a nontransparent target task has the option of accepting or explicitly rejecting a logical link request.

Receiving Connection Requests

This section describes what happens when you receive single and multiple connection requests. The remote node is assumed to be VMS. If the remote node on which your target task runs is other than VMS, you should refer to the related DECnet documentation.

When a remote node receives a call requesting a logical link, the DECnet–VAX software constructs an NCB from the information contained in the call. At this point, one of two things occurs. If a task already running on the remote node has declared a network name or object number which is the same as the one identified in the constructed NCB, the software puts the NCB into that task's mailbox. If not, DECnet–VAX must create a process to execute the task. The DECnet–VAX software either uses a compatible netserver process (if one exists) or creates a netserver process (if one does not already exist) to execute NETSERVER.COM, which in turn runs NETSERVER.EXE.

If the task running on the remote node has not declared a network name or network object, SYS$NET is equated to the NCB, and LOGIN.COM (if it exists) is invoked, which in turn starts the *taskname*.COM command file. The name of this command file is determined as follows:

- If the connection request identifies a numbered (nonzero) object, then the appropriate record is located in the configuration database and the name of the file is found in this record. (The file is assumed to reside in SYS$SYSTEM.)

- If the connection request identifies a named object with type 0 (TASK), then the file name is assumed to be the name of the task connected to (with a file type of COM) and is assumed to reside in the default directory associated with the access control information.

When executing, the target task can determine whether to accept or explicitly reject the connection request. You can program the target task to base this assessment on the information contained in the NCB.

A nontransparent target task can accept only one connection request at a time, unless it declares itself as a network task. The target task may retrieve the connection information by translating the logical name SYS$NET using the $TRNLNM system service. After the task retrieves the logical name, it may decide whether to accept or explicitly reject the connection request.

Note that you need to translate SYS$NET only if you require the following information:

- The optional data in the network connect block
- The identity of the connector
- The descriptor by which the connection was made

A target task can accept multiple inbound connection requests only if it declares itself a known network task. To make this declaration, you must first use the $ASSIGN call to assign an I/O channel to _NET:. Then, use the $QIO system service with the function code IO$_ACPCONTROL to assign a network name or object number to the task, making it eligible to process multiple inbound connection requests. This system service requires SYSNAM privilege. You must associate a mailbox with the channel if a name or number is to be declared.

You should program tasks that have declared names or object numbers to terminate when their mailboxes receive a MSG$_NETSHUT message. You must restart such tasks when the network comes back up.

After you declare the target task as an active network task, DECnet places all connection requests addressed to the task in the mailbox associated with the channel over which the ACP control function was issued. The target task retrieves connection requests from the mailbox by issuing the $QIO system service call with the function code IO$_READVBLK. Note that the first message in the mailbox is the NCB from the original connection request that put the task into a state of execution. After the task declares a network name or object number, subsequent inbound connection requests are not checked for their access control information.

Note that you can start tasks that declare names or object numbers apart from the first inbound connection (that is, by a RUN command). However, if the network task is started separately from a DECnet operation, access control is never checked.

Accepting or Rejecting a Connection Request

The target task can either accept or reject a connection request. To accept a connection request, thus completing the logical link connection, use the $QIO system service with the function code IO$_ACCESS. To reject the connection request, use the $QIO system service with the function code IO$_ACCESS!IO$M_ABORT. When it either accepts or rejects the connection request, the target task can also send 1 to 16 bytes of optional data within a modified NCB back to the source task.

Exchanging Data Messages and Interrupt Messages

The exchange of data messages between the two cooperating tasks is performed in the same way for both nontransparent and transparent communication. (Refer to Section 8.5.4.3 for information about exchanging messages on DECnet–VAX logical links.)

The exchange of interrupt messages applies only to nontransparent communication. Either task can send a 1- to 16-byte interrupt message. You can use this method to send a message to a target task outside the normal flow of data messages. DECnet–VAX places the received interrupt message in the target task's mailbox. Use the $QIO system service with the function code IO$_WRITEVBLK!IO$M_INTERRUPT to send the interrupt message. If the target task needs to be notified that an interrupt message has been placed in its mailbox, then it should issue a $QIO system service read request to the mailbox. The task may also specify an AST on the $QIO request to cause the execution of a special routine to handle the received interrupt message. (AST routines are described in the *VMS System Services Reference Manual.*)

8.6.1.6 Disconnecting or Aborting the Logical Link

A nontransparent task can terminate communication with a remote task either by disconnecting the link (synchronous disconnect or disconnect abort) or by deassigning the channel. In the first instance, you can issue a new connection request on the same channel because you do not deassign it. If you specifically use the IO$_DEACCESS, as opposed to the $DASSGN method of terminating a link, you can send an optional message of 1 to 16 bytes of data with the $QIO call.

To disconnect a logical link synchronously, issue the $QIO system service with the function code IO$_DEACCESS!IO$M_SYNCH. The channel is then free for subsequent communication with either the same or a different remote task.

A synchronous disconnect may be useful for master/slave communication, in which one task always sends data and its partner task always receives data. If the receiving task is notified of a synchronous disconnection, then all the data that was sent has been received. (The sending task, on the other hand, is not guaranteed that its partner received the data.) Because this notification is the only guarantee provided by this operation, using this operation is discouraged in favor of a user-defined protocol to ensure completion of communication. In general, the receiver of the final message should break the logical link.

To abort the logical link, issue the $QIO system service with the function code IO$_DEACCESS!IO$M_ABORT. This type of disconnect indicates that all messages transmitted by the local transmitter may not have been received or acknowledged by the remote ECL before the logical link was disconnected. You should use this type of disconnect when the local task needs to reset the logical link to a known state. If the local task needs to ensure that the transmitted messages have been received and acknowledged by the remote ECL, the task can issue the system service $CANCEL on the channel before issuing the disconnect abort. Note that this does not guarantee the delivery of the received data to the remote task. It is the responsibility of cooperating tasks to agree on a protocol to ensure that the received data is delivered to the remote task.

Note that after either a synchronous disconnect or a disconnect abort, you can issue a new connection request if you did not deassign the I/O channel.

If you issue the $CANCEL system service to a channel over which a network name or object has been declared, the declaration is removed from the network database.

8.6.1.7 Terminating the Logical Link

You can issue the $DASSGN system service call to deassign the channel and terminate the logical link immediately. You issue the call only after all communication between the tasks is complete. The call releases the I/O channel, disassociates the mailbox from the channel, and terminates the logical link immediately. This operation is equivalent to using $CANCEL followed by $QIO IO$_DEACCESS!IO$M_ABORT.

The same status and error-reporting considerations apply to nontransparent as to transparent task-to-task communication. Refer to Section 8.5.4.5 for information about status and error reporting.

8.6.2 System Service Calls for Nontransparent Operations

The following sections describe the VMS system services you can use for nontransparent task communication over the network. Each description covers the use of the call, its format, the arguments associated with the call, and the return status information. The *VMS System Messages and Recovery Procedures Reference Volume* lists the entire set of network system service error messages.

The following system services are not described in detail here, because their use does not change in a networking context. For a description of these system services, see the *VMS System Services Reference Manual.*

- $CANCEL (Cancel I/O on Channel)
- $CREMBX (Create Mailbox and Assign Channel)
- $GETDVI (Get Device/Volume Information)

Note that $GETDVI performs the same function as the Get I/O Channel Information ($GETCHN) system service. However, DIGITAL recommends that you use the $GETDVI system service.

After you issue a $CANCEL on a DECnet–VAX logical link, the only valid operation is to disconnect or abort the logical link.

8.6.2.1 $ASSIGN (I/O Channel Assignment)

The $ASSIGN system service assigns a channel to refer to a logical link. You use this channel in all $QIO calls that communicate with a remote task. In addition, you can use the $ASSIGN system service call to associate a mailbox with the channel.

Format

```
$ASSIGN devnam ,chan ,[acmode] ,[mbxnam]
```

Arguments

devnam	Address of a quadword descriptor of a character string containing the string _NET: or a logical name for _NET:.
chan	Address of a word that is to receive the assigned channel number.
acmode	Access mode to be associated with this channel. The most privileged access mode used is the access mode of the caller. You can perform I/O operations on the channel only from equal or more privileged access modes.
mbxnam	Address of a character string descriptor for the physical name of the mailbox to be associated with the channel. This mailbox remains associated with the channel until the channel is deassigned ($DASSGN).

Return Status

SS$_NORMAL	The service completed successfully.
SS$_INSFMEM	There is not enough system dynamic memory to complete the request.
SS$_NOPRIV	The issuing task does not have the required privileges to create the channel.
SS$_NOSUCHDEV	The network device driver is not loaded (for example, the DECnet–VAX software is not running currently on the local node).

8.6.2.2 $QIO (Requesting a Logical Link Connection)

The $QIO system service with the function code IO$_ACCESS requests a logical link connection to a target task. You can send 1 to 16 bytes of optional data to the target task at the same time that you issue the $QIO system service.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,[p1] ,p2

Arguments

efn	Number of the event flag to be set at request completion.
chan	Channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_ACCESS.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of an AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Not used (omit the argument).
p2	Address of a quadword descriptor of the NCB (see Section 8.6.1.4). Both the descriptor and the NCB must be in read/write storage.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_CONNECFAIL	The connection to a network object timed out or failed.
SS$_DEVOFFLINE	The physical link is shutting down.
SS$_FILALRACC	A logical link is already accessed on the channel (that is, a previous connection is active on the channel).
SS$_INSFMEM	There is not enough system dynamic memory to complete the request.
SS$_INVLOGIN	The access control information was found to be invalid at the remote node.
SS$_IVDEVNAM	The NCB has an invalid format or content.
SS$_LINKEXIT	The network partner task was started, but exited before confirming the logical link (that is, $ASSIGN to SYS$NET).
SS$_NOLINKS	No logical links are available. The maximum number of logical links as set for the executor MAXIMUM LINKS parameter was exceeded.
SS$_NOPRIV	The issuing task does not have the required privileges to create a logical link to the designated target.
SS$_NOSUCHNODE	The specified node is unknown.
SS$_NOSUCHOBJ	The network object number is unknown at the remote node; or for a TASK= connect, the named DCL command procedure file cannot be found at the remote node.
SS$_NOSUCHUSER	The remote node could not recognize the login information supplied with the connection request.
SS$_PROTOCOL	A network protocol error occurred. This error is most likely due to a network software error.
SS$_REJECT	The network object rejected the connection.
SS$_REMRSRC	The link could not be established because system resources at the remote node were insufficient.
SS$_SHUT	The local or remote node is no longer accepting connections.
SS$_THIRDPARTY	The logical link was terminated by a third party (for example, the system manager).
SS$_TOOMUCHDATA	The task specified too much optional or interrupt data.
SS$_UNREACHABLE	The remote node is currently unreachable.

8.6.2.3 $QIO (Accepting Logical Link Connection Request)

The $QIO system service with the function code IO$_ACCESS accepts a logical link connection request from a source task. You can send 1 to 16 bytes of optional data to the source task at the same time that you issue the $QIO system service.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,[p1] ,p2

Arguments

efn	Number of the event flag to be set at request completion.
chan	Channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_ACCESS.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of an AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Not used (omit the argument).
p2	Address of a quadword descriptor of the NCB (see Section 8.6.1.4). Both the descriptor and the NCB must be in read/write storage.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_DEVALLOC	The process cannot access the logical link specified in the NCB because that link is intended for another process.
SS$_EXQUOTA	The process does not have sufficient quota to complete the request.
SS$_INSFMEM	There is not enough system dynamic memory to complete the request.
SS$_IVDEVNAM	The NCB has an invalid format or content.
SS$_LINKABORT	The network partner task aborted the logical link.
SS$_LINKDISCON	The network partner task disconnected the logical link.
SS$_LINKEXIT	The network partner task exited.
SS$_NOSUCHNODE	The specified node is unknown.
SS$_PATHLOST	The path to the network partner task node was lost.
SS$_PROTOCOL	A network protocol error occurred. This error is most likely due to a network software error.

SS$_THIRDPARTY	The logical link connection was terminated by a third party (for example, the system manager).
SS$_TIMEOUT	The connection request did not complete within the required time.
SS$_UNREACHABLE	The remote node is currently unreachable.

8.6.2.4 $QIO (Rejecting a Logical Link Connection Request)

The $QIO system service with the function code IO$_ACCESS!IO$M_ABORT rejects a logical link connection request. You can send 1 to 16 bytes of optional data to the target task at the same time that you issue the $QIO system service.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,[p1] ,p2

Arguments

efn	Number of the event flag to be set at request completion.
chan	Channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_ACCESS!IO$M_ABORT.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of an AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Not used (omit the argument).
p2	Address of a quadword descriptor of the NCB (see Section 8.6.1.4). Both the descriptor and the NCB must be in read/write storage.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_DEVALLOC	The process cannot access the logical link specified in the NCB because that link is intended for another process.
SS$_EXQUOTA	The process does not have sufficient quota to complete the request.
SS$_IVDEVNAM	The NCB has an invalid format or content.
SS$_LINKABORT	The network partner task aborted the logical link.
SS$_LINKDISCON	The network partner task disconnected the logical link.
SS$_LINKEXIT	The network partner task exited.
SS$_NOSUCHNODE	The specified node is unknown.
SS$_TIMEOUT	The connection request did not complete within the required time.
SS$_PATHLOST	The path to the network partner task node was lost.

SS$_PROTOCOL	A network protocol error occurred. This error is most likely due to a network software error.
SS$_THIRDPARTY	The logical link connection was terminated by a third party (for example, the system manager).
SS$_UNREACHABLE	The remote node is currently unreachable.

8.6.2.5 $QIO (Sending a Message to a Target Task)

The $QIO system service with the function code IO$_WRITEVBLK or IO$_WRITEVBLK!IO$M_INTERRUPT or IO$_WRITEVBLK!IO$M_MULTIPLE sends a message to a target task. Refer to Section 8.5.5.2 for the format of this call, its arguments, and possible return status codes.

8.6.2.6 $QIO (Receiving a Message from a Target Task)

The $QIO system service with the function code IO$_READVBLK or IO$_READVBLK!IO$M_MULTIPLE receives a message from a target task. Refer to Section 8.5.5.3 for the format of this call, its arguments, and possible return status codes.

8.6.2.7 $QIO (Sending an Interrupt Message to a Target Task)

The $QIO system service with the function code IO$_WRITEVBLK!IO$M_INTERRUPT sends a 1- to 16-byte interrupt message to a target task. If the remote node is a VMS operating system, the message is placed in the mailbox associated with the target task.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,p1 ,p2

Arguments

efn	Number of the event flag to be set at event completion.
chan	Channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_WRITEVBLK!IO$M_INTERRUPT.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of the AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Buffer address.
p2	Buffer length (1 to 16 bytes).

Return Status

SS$_NORMAL	The service completed successfully.
SS$_ABORT	The I/O request has been aborted by a $DASSGN or $CANCEL call.
SS$_FILNOTACC	No logical link is associated with the channel.

SS$_INSFMEM	Enough memory to buffer the message could not be allocated.
SS$_LINKABORT	The network partner task aborted the logical link.
SS$_LINKDISCON	The network partner task disconnected the logical link.
SS$_LINKEXIT	The network partner task exited.
SS$_NOSOLICIT	DECnet could not accept an interrupt message at this time.
SS$_TOOMUCHDATA	The task specified too much interrupt data.
SS$_PATHLOST	The path to the network partner task node was lost.
SS$_PROTOCOL	A network protocol error occurred. This error is most likely due to a network software error.
SS$_THIRDPARTY	The logical link connection was terminated by a third party (for example, the system manager).

8.6.2.8 $QIO (Synchronously Disconnecting a Logical Link)

The $QIO system service with the function code IO$_DEACCESS!IO$M_SYNCH synchronously disconnects the logical link. All pending messages are transmitted to the remote node before the link is disconnected.

You can send 1 to 16 bytes of optional data to the task from which you are disconnecting at the same time you issue this $QIO system service.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,[p1] ,[p2]

Arguments

efn	Number of the event flag to be set at event completion.
chan	Channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_DEACCESS!IO$M_SYNCH.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of the AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Not used (omit the argument).
p2	Address of a descriptor of a counted ASCII string of optional user data. Both the string and its descriptor must be in read/write storage.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_FILNOTACC	No logical link is associated with the channel.

8.6.2.9 $QIO (Aborting a Logical Link)

The $QIO system service with the function code IO$_DEACCESS!IO$_ABORT terminates the logical link. Note, however, that the DEACCESS function completes only after all pending I/O operations complete, even if you specify IO$_ABORT. First, issue the $CANCEL system service call to cancel I/O operations on the logical link and then issue this call to terminate the logical link.

You can send 1 to 16 bytes of optional data to the task from which you are disconnecting at the same time that you issue this $QIO system service call.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,[p1] ,[p2]

Arguments

efn	Number of the event flag to be set at event completion.
chan	Channel number associated with the logical link. Use the same channel number returned in the $ASSIGN call.
func	IO$_DEACCESS!IO$M_ABORT.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of the AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Not used (omit the argument).
p2	Address of a quadword descriptor of a counted string of optional user data. Both the string and its descriptor must be in read/write storage.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_FILNOTACC	No logical link is associated with the channel.

8.6.2.10 $QIO (Declaring a Network Name or Object Number)

The $QIO system service with the function code IO$_ACPCONTROL assigns a network name or object number to the task, thereby making it eligible to process multiple inbound connection requests. You must associate a mailbox with the I/O channel. All inbound connection requests are placed in the mailbox associated with the channel over which this I/O function is issued. You need the SYSNAM privilege to declare a name or object number.

MACRO programmers should be aware that, whenever a logical link is established, you should obtain its device unit number (for example, 18 from _NET18:) by using the $GETDVI system service, because unit numbers and not channel numbers appear in mailbox messages. Use this system service call where a single mailbox is being used for many logical links. The unit number could be used as a key into a database that keeps track of multiple links.

Format

$QIO [efn] ,chan ,func ,[iosb] ,[astadr] ,[astprm] ,p1 ,p2

Arguments

efn	Number of the event flag to be set at event completion.
chan	Word containing the channel number associated with the logical link. Use the same channel number assigned in the $ASSIGN call.
func	IO$_ACPCONTROL.
iosb	Address of a quadword I/O status block that is to receive the completion status.
astadr	Entry point address of the AST routine that executes when the I/O operation completes. If specified, the AST routine executes at the access mode from which the $QIO service was requested.
astprm	AST parameter to be passed to the AST completion routine.
p1	Address of a quadword descriptor of a 5-byte block consisting of a function type (one byte) and a longword parameter. The function type is a symbol defined by the $NFBDEF macro in SYS$LIBRARY:LIB.MLB. The format of the 5-byte block for declaring a name is as follows: .BYTE NFB$C_DECLNAME .LONG 0 The format of the 5-byte block for declaring an object number is as follows: .BYTE NFB$C_DECLOBJ .LONG object-number The object number is a number reserved for customer use in the range of 128 to 255. This 5-byte buffer and its descriptor should be in read/write storage.
p2	Address of a quadword descriptor of the network name (maximum of 12 characters). You should not supply this argument for the DECLOBJ function. Both the name and its descriptor must be in read/write storage.

Return Status

SS$_NORMAL	The service completed successfully.
SS$_BADPARAM	One of the QIO parameters has an invalid value.
SS$_ILLCNTRFUNC	The control function is invalid.
SS$_NOMBX	A name or object number is being declared using a channel without an associated mailbox.
SS$_NOPRIV	The issuing process does not have the SYSNAM privilege.

8.6.2.11 $DASSGN (Terminating a Logical Link)

The $DASSGN system service terminates all pending operations to send and receive data, aborts the logical link immediately, and frees the channel associated with that link. Refer to Section 8.5.5.4 for the format of this call, its arguments, and possible return status codes.

8.7 Designing Tasks

The following sections contain a command procedure and three user program examples designed to perform task-to-task communications over the network.

The command procedure and the first two user program examples illustrate transparent operations. The third user program example illustrates a nontransparent operation.

8.7.1 DCL Command Procedure for Task-to-Task Communication

As described in Section 8.5, you can write command procedures in DCL to execute transparent task-to-task operations. You can use the following command procedure, called SHOWBQ.COM, to perform such an operation. You can use SHOWBQ.COM for task-to-task communication by entering a task specification string in a TYPE command. For example:

```
$ TYPE TRNTO"BROWN JUNE"::"TASK=SHOWBQ"
```

In this command procedure, SYS$OUTPUT is equated to SYS$NET in user mode to allow the SHOW QUEUE image to communicate over the logical link by opening SYS$OUTPUT. When the SHOW QUEUE image exits, the temporary definition of SYS$OUTPUT is deleted. In other words, only one DCL image can use the logical link as the communication path to the requester at the other node.

```
        SHOWBQ.COM
$ !
$ !     This  command procedure  returns status information about
$ !     jobs  entered in  batch  queues  on  the  system where it
$ !     executes.   It  may  be   run  interactively as  a command
$ !     procedure,  submitted  as a  local or remote batch job, or
$ !     invoked  as a  "remote task"  to display information about
$ !     another system.  For example:
$ !
$ !     $ @SHOWBQ
$ !     $ SUBMIT SHOWBQ
$ !     $ SUBMIT/REMOTE node::SHOWBQ
$ !     $ TYPE node::"TASK=SHOWBQ"
$ !
$ IF F$MODE() .EQS. "NETWORK" THEN GOTO NET
$ SHOW QUEUE/BATCH/BRIEF/ALL
$ EXIT
$NET:
$ DEFINE/USER SYS$OUTPUT SYS$NET
$ SHOW QUEUE/BATCH/BRIEF/ALL
$ EXIT
```

8.7.2 FORTRAN Program for Task-to-Task Communication

Example 8–2 shows an example of VAX FORTRAN transparent communication. In the FORTRAN source task that initiates the logical link request, you use a standard open statement to specify the remote task to which you want to connect. In turn, the remote task issues an open statement to complete the logical link connection. A coordinated set of read and write operations enable the exchange of information over the link. To terminate the connection, the source task executes a close statement to break the logical link. When the remote task receives this close statement, it issues a close statement, which completes the link termination process. The remainder of this section discusses the statements that you would use to develop such an application.

Example 8–2 FORTRAN Task-to-Task Communication

```
        PROGRAM TEST3.FOR
C
C       This program prompts the user for the part number of an item
C       in inventory and responds with the number of units in stock.
C       The information is obtained by communicating with a program
C       (TEST4) on another node that has access to the inventory data.
C
C       Before running this program, the logical name TASK must be
C       defined to refer to the target program. For example:
C
C       $ DEFINE TASK "TRNTO::""TASK=TEST4"""
C       $ RUN TEST3
C
        CHARACTER PARTNO*5
        INTEGER PARTCOUNT

C
100     FORMAT (/,'$Enter part number: ')
200     FORMAT (A)
300     FORMAT (I4)
400     FORMAT ('0Inventory for part number ',A,' is: ',I4)
C
C       Establish a logical link with the target task.
C
   ❶  OPEN (UNIT=1,NAME='TASK',ACCESS='SEQUENTIAL',
     1       FORM='FORMATTED',CARRIAGECONTROL='NONE',TYPE='NEW')
C
C       Prompt the user for a part number, send it to the target task,
C       read reply of quantity on hand, and finally display the answer
C       for the user. Repeat the cycle until the user enters 'EXIT' for
C       a part number.
C
10      TYPE 100
        ACCEPT 200, PARTNO
        IF (PARTNO .EQ. 'EXIT') GOTO 20
    ❷  WRITE (1,200) PARTNO
        READ (1,300) PARTCOUNT
        TYPE 400, PARTNO, PARTCOUNT
        GOTO 10
C
C       Disconnect the logical link.
C
20  ❸  CLOSE (UNIT=1)
```

Example 8–2 Cont'd. on next page

Example 8–2 (Cont.) FORTRAN Task-to-Task Communication

```
        END
$!
$ !     ********************************************************
$ !     TEST4.COM
$ !
$ !     This command procedure executes the program TEST4 after
$ !     being started by a task-to-task connection request issued
$ !     by TEST3.
$ !
❹$ RUN SYS$LOGIN:TEST4.EXE
$ EXIT
        PROGRAM TEST4.FOR
C
C       Test4 is the target program that communicates with TEST3.
C       For each part number received from the source task, the
C       number of units in stock is determined, and this value is
C       returned.
C
C       To complete the logical link with its initiator, this program
C       uses the logical name SYS$NET as the file specification in an
C       open statement.
C

        CHARACTER PARTNO*5
        INTEGER PARTCOUNT
C
100     FORMAT (A)
200     FORMAT (I4)
C
C       Complete the logical link connection.
C
    ❺ OPEN (UNIT=1,NAME='SYS$NET',ACCESS='SEQUENTIAL',
     1       FORM='FORMATTED',CARRIAGECONTROL='NONE',TYPE='OLD')
C
C       Process requests until end-of-file is reached. (This is the
C       error condition returned when the source task breaks the
C       logical link connection.)
C
10  ❷ READ    (1,100,END=20) PARTNO
C
C       Perform appropriate processing to obtain the part count value
C       and transmit this back to the source task.
C
        CALL INSTOCK (PARTNO,PARTCOUNT)
    ❷ WRITE (1,200) PARTCOUNT
        GOTO 10

C
C       Disconnect the logical link.
C
20  ❸ CLOSE   (UNIT=1)
        END
```

Notes on Example 8–2

❶ The source task, TEST3, requests a logical link connection to the target task, TEST4.

❷ TEST3 and TEST4 send and receive data messages.

❸ TEST3 disconnects the logical link and thereby terminates the communication process.

❹ When the remote node receives a connection request, the command procedure TEST4.COM is executed. This command procedure must reside in the default directory associated with the account being accessed. TEST4.COM contains a RUN statement that causes the program TEST4.EXE to be executed.

❺ TEST4 completes the logical link connection through SYS$NET. Note that the unit numbers in the source and target programs need not be the same.

Because DECnet–VAX translates system-dependent language calls into the same set of messages that permit task-to-task communication, any task programmed in VAX MACRO or one of the higher-level languages can communicate with a remote task programmed in the same or a different language. More specifically, for communication between tasks that run on VMS nodes, the language in which you access the network has no effect on the communication process. The VAX FORTRAN source task in Example 8–2 could just as easily communicate with a MACRO task on node TRNTO.

8.7.3 MACRO Program for Transparent Task-to-Task Communication

Example 8–3 illustrates the use of system service calls for transparent communication. TRANA is a MACRO source task on the local node that communicates with a target task, TRANB, on node TRNTO. The source task sends a connection request to the remote node whereupon the target task is started by the command file TRANB.COM. After the logical link connection is made, the source task sends a message to the target task, which in turn responds with a message and then waits for additional message traffic. The source task drives the communication process. After the source task receives a response from the target task, it disconnects the link and exits, which causes the target task to exit also, thereby terminating the communication process.

Example 8–3 Transparent Communication Using System Services

```
        .TITLE  TRANA   - SOURCE TASK USING TRANSPARENT I/O
        .IDENT  /V1.0/
        .SBTTL  WRITABLE_DATA
        .PSECT  TRANA$DATA      SHR,NOEXE,RD,WRT,BYTE

NETCHAN:.BLKW   1                                       ; Network channel
IOSBUF: .BLKQ   1                                       ; I/O status block
TARGET: .ASCID  /TRNTO"MALIK KARL"::"TASK=TRANB"/       ; Task-spec (and descriptor)

SENDMSG:.ASCII  /SEND THIS STRING TO TRANB/             ; Output buffer
SENDMSG_SIZ=.-SENDMSG                                   ; Output buffer size
RECVMSG:.BLKB   512                                     ; Input buffer
RECVMSG_SIZ=.-RECVMSG                                   ; Input buffer size
```

Example 8–3 Cont'd. on next page

Example 8–3 (Cont.) Transparent Communication Using System Services

```
        .SBTTL  MAIN
        .PSECT  TRANA$CODE      NOSHR,EXE,RD,NOWRT,BYTE
        .ENTRY  START,^M<>              ; Entry point from exec
;
; Request a logical link to the target task and assign an I/O channel.
;
        $ASSIGN_S -                     ; Assign a channel to target task
                DEVNAM=W^TARGET,-       ; Address of device name descriptor
                CHAN=W^NETCHAN          ; Address to receive channel number
        BLBC    R0,EXIT                 ; Branch on failure
;
; Send a message to the target task.
;
        $QIOW_S -                       ; Issue transmit request
                EFN=#1,-                ; Use local event flag number 1
                CHAN=W^NETCHAN,-        ; Use assigned channel
                FUNC=S^#IO$_WRITEVBLK,- ; Write virtual block
                IOSB=W^IOSBUF,-         ; Address of I/O status block
                P1=W^SENDMSG,-          ; Address of output buffer
                P2=S^#SENDMSG_SIZ       ; Size of output buffer
        BLBC    R0,EXIT                 ; Branch on failure
        MOVZWL  W^IOSBUF,R0             ; Get completion status
        BLBC    R0,EXIT                 ; Branch on failure
;
; Receive a message from the target task.
;
        $QIOW_S -                       ; Issue receive request
                EFN=#1,-                ; Use local event flag number 1
                CHAN=W^NETCHAN,-        ; Use assigned channel
                FUNC=S^#IO$_READVBLK,-  ; Read virtual block
                IOSB=W^IOSBUF,-         ; Address of I/O status block
                P1=W^RECVMSG,-          ; Address of input buffer
                P2=#RECVMSG_SIZ         ; Size of input buffer
        BLBC    R0,EXIT                 ; Branch on failure
        MOVZWL  W^IOSBUF,R0             ; Get completion status
        BLBC    R0,EXIT                 ; Branch on failure
;
; Abort the logical link.
;
        $DASSGN_S -                     ; Deassign the channel
                CHAN=W^NETCHAN          ; Address of word containing channel number
;
; Exit with status (in R0).
;
EXIT:   $EXIT_S R0                      ; Exit with status to be displayed
                                        ;  on error condition

        .END    START                   ; Image transfer address
;
;
;*****************************************************************
        .TITLE  TRANB - TARGET TASK USING TRANSPARENT I/O
        .IDENT  /V1.0/
        .SBTTL  WRITABLE_DATA
        .PSECT  TRANB$DATA      SHR,NOEXE,RD,WRT,BYTE
```

Example 8–3 Cont'd. on next page

Example 8–3 (Cont.) Transparent Communication Using System Services

```
NETCHAN:.BLKW   1                       ; Network channel
IOSBUF: .BLKQ   1                       ; I/O status block
RECVMSG:.BLKB   512                     ; Input buffer
RECVMSG_SIZ=.-RECVMSG                   ; Input buffer size
LOGNAM: .ASCID  /SYS$NET/               ; Logical name and descriptor
SENDMSG:.ASCII  /REPLY TO TRANA/        ; Output buffer
SENDMSG_SIZ=.-SENDMSG                   ; Output buffer size

        .SBTTL  MAINLINE
        .PSECT  TRANB$CODE      NOSHR,EXE,RD,NOWRT,BYTE
        .ENTRY  START,^M<>              ; Entry point from exec
;
; Complete the logical link connection (that TRANA requested).
;
        $ASSIGN_S -                     ; Assign channel to SYS$NET
                DEVNAM=W^LOGNAM,-       ; Descriptor of SYS$NET
                CHAN=W^NETCHAN          ; Store channel number
        BLBC    R0,EXIT                 ; Branch on failure
LOOP:
;
; Receive message from source task.
;
        $QIOW_S -                       ; Issue receive request
                EFN=#1,-                ; Use local event flag number 1
                CHAN=W^NETCHAN,-        ; Use assigned channel
                FUNC=S^#IO$_READVBLK,-  ; Read virtual block
                IOSB=W^IOSBUF,-         ; Address of I/O status block
                P1=W^RECVMSG,-          ; Address of input buffer
                P2=#RECVMSG_SIZ         ; Size of input buffer
        BLBC    R0,EXIT                 ; Branch on failure
        MOVZWL  W^IOSBUF,R0             ; Get completion status
        CMPW    S^#SS$_ABORT,R0         ; Was logical link aborted?
        BEQL    DONE                    ; Branch if yes
        BLBC    R0,EXIT                 ; Branch on failure
;
; Send message to source task.
;
        $QIOW_S -                       ; Issue transmit request
                EFN=#1,-                ; Use local event flag number 1
                CHAN=W^NETCHAN,-        ; Use assigned channel
                FUNC=S^#IO$_WRITEVBLK,- ; Write virtual block
                IOSB=W^IOSBUF,-         ; Address of I/O status block
                P1=W^SENDMSG,-          ; Address of output buffer
                P2=S^#SENDMSG_SIZ       ; Size of output buffer
        BLBC    R0,EXIT                 ; Branch on failure
        MOVZWL  W^IOSBUF,R0             ; Get completion status
        BLBC    R0,EXIT                 ; Branch on failure
        BRB     LOOP                    ; Reissue the read
;
; Logical link was aborted.
;
DONE:   $DASSGN_S -                     ; Deassign the channel
                CHAN=W^NETCHAN          ; Address of channel number
```

Example 8–3 Cont'd. on next page

Example 8–3 (Cont.) Transparent Communication Using System Services

```
;
; Exit with status (in R0).
;
EXIT:   $EXIT_S R0                      ; Exit with status to be displayed
                                        ;  on error condition

        .END    START                   ; Image transfer address
```

8.7.4 MACRO Program for Nontransparent Task-to-Task Communication

Example 8–4 illustrates the use of several system service calls for nontransparent task-to-task communication. DB_REQUESTER is a nontransparent MACRO source task on the local node that communicates with a nontransparent target task, DB_SERVER, on node BIGRED. The task DB_SERVER executes a database inquiry at the target node using the key information that is input at the originator node. This example is similar to Example 8–3, except that the source task here uses a network connect block (NCB) and performs a nontransparent assign operation to establish communication with the target task. DB_SERVER is a nontransparent target task that has declared a name (that is, it is eligible to receive multiple inbound connection requests). In addition, it uses a mailbox to receive network status notifications.

The programs shown in Example 8–4 are available on the VMS distribution medium. To access the programs, specify the file names SYS$EXAMPLES:DB_REQUESTER.MAR and SYS$EXAMPLES:DB_SERVER.MAR.

Example 8–4 Nontransparent Communication Using System Services

```
        .TITLE  DB_REQUESTER -    Database request program
        .IDENT  /V1.0/
        .SBTTL  DEFINITIONS

;+
;  This program demonstrates how to perform task-to-task communication
;  with the known network object, DB_SERVER.  A database inquiry is
;  executed by the DB_SERVER process at the target node using key
;  information (name) input on the originating node.
;-

        .DSABL  GLOBAL

;+
;  Include system macros for definition
;-

$DSCDEF                                 ; Descriptor definitions
$IODEF                                  ; I/O function codes
$RMSDEF                                 ; RMS status values
$SSDEF                                  ; System status values
;+
;  Local definitions
;-
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
MAX_MSG = 128                           ; _NET: mailbox size
BUF_QUO = 128                           ; Only one message

$DEFINI BUFF_DEF                        ; Buffer layout
        $DEF    BUFF_T_NAME     .BLKB   20
        $EQU    BUFF_S_NAME     < . - BUFF_T_NAME>
        $DEF    BUFF_T_ACCOUNT  .BLKB   11
        $EQU    BUFF_S_ACCOUNT  < . - BUFF_T_ACCOUNT>
        $DEF    BUFF_T_PHONE    .BLKB   14
        $EQU    BUFF_S_PHONE    < . - BUFF_T_PHONE>
        $DEF    BUFF_T_ADDRESS   .BLKB  30
        $EQU    BUFF_S_ADDRESS  < . - BUFF_T_ADDRESS>
        $DEF    BUFF_T_LOCATION .BLKB   30
        $EQU    BUFF_S_LOCATION < . - BUFF_T_LOCATION>
        $DEF    BUFF_L_STATUS   .BLKL    1
        $DEF    BUFF_T_SPARE    .BLKB    7
        $EQU    BUFF_S_SPARE    < . - BUFF_T_SPARE>
        $DEF    BUFF_K_LEN
$DEFEND BUFF_DEF

;+
;  Declare external routines
;-

        .EXTRN  LIB$ASN_WTH_MBX,-       ; Assign a channel and associate a mailbox with it
                LIB$GET_INPUT,-         ; Get input from SYS$INPUT
                LIB$PUT_OUTPUT          ; Write output to SYS$OUTPUT

        .SBTTL  RO_DATA - Read Only DATA
        .PSECT  RO_DATA RD,NOWRT,NOEXE

INPUT_PROMPT:   .ASCID  /Input name or ^Z to exit : /

FAO_CTRL:       .ASCID  "!/!AF Account: !AF  Phone: !AF!/  Address: !AF"-
                        "  City: !AF!/"

NET_DEVICE:     .ASCID  /_NET:/

        .SBTTL  RW_DATA - Read Write DATA
        .PSECT  RW_DATA RD,WRT,NOEXE

IOSB:           .BLKQ                   ; I/O status block

MSG_VEC:                                ; Message vector
                .WORD   1               ;   Count of vector items
                .WORD   15              ;   All options on (FAC, SEV, IDT, TXT)
CODE:           .BLKL   1               ;   Message code

FAO_PRMLST:     .LONG    BUFF_S_NAME    ; Display matches FAO_CTRL
                .ADDRESS BUFFER+BUFF_T_NAME
                .LONG    BUFF_S_ACCOUNT
                .ADDRESS BUFFER+BUFF_T_ACCOUNT
                .LONG    BUFF_S_PHONE
                .ADDRESS BUFFER+BUFF_T_PHONE
                .LONG    BUFF_S_ADDRESS
                .ADDRESS BUFFER+BUFF_T_ADDRESS
                .LONG    BUFF_S_LOCATION
                .ADDRESS BUFFER+BUFF_T_LOCATION

NET_CHAN:       .BLKW   1               ; Channel to _NET: device
MBX_CHAN:       .BLKW   1               ; Channel to associated mailbox
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
DISP_DESC:      .WORD   MAX_DISPLAY     ;   Length
                .BYTE   DSC$K_DTYPE_T   ;   Text
                .BYTE   DSC$K_CLASS_S   ;   String
                .ADDRESS DISPLAY_BUFF   ;   Pointer

DISPLAY_BUFF:   .BLKB   160             ; Display buffer
MAX_DISPLAY = < . - DISPLAY_BUFF>

BUFFER:         .BLKB   BUFF_K_LEN      ; Buffer for link communication

INQ_NAME:       .WORD   BUFF_S_NAME     ; Descriptor of input name key
                .WORD   0
                .ADDRESS BUFFER+BUFF_T_NAME

NCB_DESC:       .WORD  NCB_LEN          ; Descriptor of network connect block (NCB)
                .WORD   0
                .ADDRESS NCB

NCB:            .ASCII  ?BIGRED::"TASK=DB_SERVER/?
                .WORD   0
;+
;  Up to 16 bytes of optional data may be sent in the connect
;  initiate request.  It is in the form of an ASCIC string and
;  is placed after the zero word and before the /"/ terminator.
-
                .ASCII  /"/
NCB_LEN = < . - NCB>

        .SBTTL  CODE - Start of program
        .PSECT  CODE    RD,NOWRT,EXE
        .ENTRY  DB_REQUESTER      ^M<>

;+
;  After initialization, the user's requests are processed until ^Z is
;  entered at the name prompt.  Processing is synchronous.  After each
;  name input, a request is sent to DB_SERVER over the logical link.
;  The program then waits until a response is received and displayed
;  before requesting another name.
;-
        BSBW    INITIALIZATION          ; Initialize communication to DB_SERVER
        BLBC    R0,99$                  ; If BC, error, return status
                                        ; While success loop
10$:    BSBW    INQUIRE_NAME            ;   Input a request
        BLBS    R0,20$                  ;   If BS, success, continue
        CMPL    #RMS$_EOF,R0            ;   End of processing?
        BNEQ    99$                     ;     If NEQ, unrecoverable error
        MOVL    #SS$_NORMAL,R0          ;     Recoverable, reset status
        BRB     99$                     ;       and return
20$:    BSBW    ISSUE_REQUEST           ;   Send the request off
        BLBC    R0,99$                  ;   If BC, error, return status
        BSBW    RCV_AND_DISP_RESPONSE   ;   Receive and display response
        BLBS    R0,10$                  ; If BS, no error, next
                                        ; We should $DASSGN channel
                                        ;   However, on image exit, all channels
                                        ;   including NET_CHAN are deassigned
99$:    RET


INITIALIZATION:
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
;+
;  Set up communications with the DB_SERVER process.
;
;  Requesting a logical link over DECnet using nontransparent communications
;  requires three steps.
;     1.  Create a temporary mailbox (optional)
;     2.  Assign a channel to the _NET: device and associate the temporary
;         mailbox with it
;     3.  Issue a QIO with a function code of IO$_ACCESS and the P2
;         parameter referencing the address of a descriptor containing the
;         network connect block (NCB).
;
;  In this example steps 1 and 2 are combined by using the RTL routine
;  LIB$ASN_WTH_MBX.
;-
        MOVAB   BUFFER,R2               ; Set base of buffer for later
        PUSHAW  MBX_CHAN                ;   Associated mailbox channel
        PUSHAW  NET_CHAN                ;   Net device channel
        PUSHAL  #BUF_QUO                ;   Buffered quota
        PUSHAL  #MAX_MSG                ;   Maximum message
        PUSHAQ  NET_DEVICE              ;   _NET: device
        CALLS   #5,G^LIB$ASN_WTH_MBX    ; Assign a channel with associated mailbox
        BLBC    R0,99$                  ; If BC, error, return status
        $QIOW_S -                       ; Issue connect initiate
                CHAN=NET_CHAN,-         ;   Net channel
                FUNC=#IO$_ACCESS,-      ;   Function
                IOSB=IOSB,-             ;   I/O status block
                P2=#NCB_DESC            ;   NCB descriptor
        BLBC    R0,99$                  ; If BC, error, return status
        MOVZWL  IOSB,R0                 ; Get completion status
99$:    RSB

INQUIRE_NAME:

;  Prompt the user for the name to query

        PUSHAQ  INPUT_PROMPT            ;   Prompt user
        PUSHAQ  INQ_NAME                ;   Inquire name
        CALLS   #2,G^LIB$GET_INPUT      ; Get input key name
        RSB

ISSUE_REQUEST:

;  Issue a write over the logical link

        $QIOW_S -                       ; Send a buffer over the logical link
                CHAN=NET_CHAN,-         ;   LL channel
                FUNC=#IO$_WRITEVBLK,-   ;   Write operation
                IOSB=IOSB,-             ;   I/O status block
                P1=BUFF_T_NAME(R2),-    ;   Buffer to send
                P2=#BUFF_S_NAME         ;   Size of name
        BLBC    R0,99$                  ; If BC, error, return status
        MOVZWL  IOSB,R0                 ; Get completion status
99$:    RSB

RCV_AND_DISP_RESPONSE:
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
;+
;  This module waits on a read over the logical link to DB_SERVER for a
;  response.  After the response is received, the buffer is formatted and
;  the information is displayed to the user's terminal.  If the server
;  encountered any errors on the request, the status in the buffer will
;  reflect the condition.
;-
        $QIOW_S  -                      ; Wait on response message
                CHAN=NET_CHAN,-         ;   LL channel
                FUNC=#IO$_READVBLK,-    ;   Read operation
                IOSB=IOSB,-             ;   I/O status block
                P1=(R2),-               ;   Buffer
                P2=#BUFF_K_LEN          ;   Length of buffer
        BLBC    R0,99$                  ; If BC, error, return status
        BLBS    IOSB,10$                ; If BS, completion OK, continue
        MOVZWL  IOSB,R0                 ; Capture error
        BRB     99$                     ;   and return status

10$:    BLBS    BUFF_L_STATUS(R2),20$   ; If BS, request successful
        MOVL    BUFF_L_STATUS(R2),CODE  ;   Set code for error message
        $PUTMSG_S -                     ;   Display the error message
                MSGVEC=MSG_VEC          ;     Message vector
        BRB     99$                     ;   Return with status

20$:    MOVW    #MAX_DISPLAY,-          ; Reset the
                DISP_DESC+DSC$W_LENGTH  ;    output length
        $FAOL_S -                       ; Format the return information
                CTRSTR=FAO_CTRL,-       ;   Control string
                OUTLEN=DISP_DESC,-      ;   Output length
                OUTBUF=DISP_DESC,-      ;   Output string descriptor
                PRMLST=FAO_PRMLST       ;   Parameter list
        BLBC    R0,99$                  ; If BC, error, return status
        PUSHAQ  DISP_DESC               ;   Formatted buffer
        CALLS   #1,G^LIB$PUT_OUTPUT     ; Display the information
99$:    RSB

       .END     DB_REQUESTER

        .TITLE  DB_SERVER - Database server process
        .IDENT  /V1.0/
        .SBTTL  DEFINITIONS

;+
;  This program demonstrates how to declare a known network object and
;  service  and manage multiple logical links.  The database USER_DB
;  is accessed based on the name key supplied in the request buffer.
;-

        .DSABL  GLOBAL
        .LIBRARY        /SYS$LIBRARY:LIB.MLB/

;+
;  Define macros,  Determine FATAL from normal DECnet error messages
;-
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
        .MACRO  BR_FATIO  DEST            ; Fatal I/O errors
        CMPL    R0,#SS$_BUFFEROVF
        BEQL    DEST
        CMPL    R0,#SS$_FILNOTACC
        BEQL    DEST
        CMPL    R0,#SS$_INSFMEM
        BEQL    DEST
        .ENDM   BR_FATIO

        .MACRO  BR_FATACC  DEST         ; Fatal connect initiate accept errors
        CMPL    R0,#SS$_DEVALLOC
        BEQL    DEST
        CMPL    R0,#SS$_INSFMEM
        BEQL    DEST
        CMPL    R0,#SS$_IVDEVNAM
        BEQL    DEST
        .ENDM  BR_FATACC

;+
;  Include system macros for definitions
;-

$DSCDEF                                 ; Descriptor definitions
$DVIDEF                                 ; GETDVI definitions
$IODEF                                  ; I/O function definitions
$MSGDEF                                 ; Message definitions (mailbox)
$NFBDEF                                 ; Network function definitions
$RMSDEF                                 ; RMS status values
$SSDEF                                  ; System status values
;+
;  Local definitions
;-

TEMP_MBX = 0                            ; Temporary mailbox
MAX_BUFFS = 100                         ; Maximum number of buffers
MAX_LINKS = 32                          ; Maximum number of logical links (<= 32)
MAX_MSG = 128                           ; Maximum mailbox message (NETCMD)
BUF_QUO = 128                           ; Only one message
MAX_NCB = 110                           ; Maximum NCB message size
NET_RD = 1                              ; Input completion on logical link
NET_WRT = 2                             ; Output completion on logical link
NET_CMD = 3                             ; Input completion on net command mailbox
FREE_QUE = 0                            ; Free queue (idle buffers)
LIVE_QUE = 1                            ; Live queue (messages to process)
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
$DEFINI DATA_DEF                        ; Data buffer/file record layout
        $DEF    DATA_T_NAME       .BLKB  20
        $EQU    DATA_S_NAME       < . - DATA_T_NAME>
        $DEF    DATA_T_ACCOUNT    .BLKB  11
        $EQU    DATA_S_ACCOUNT    < . - DATA_T_ACCOUNT>
        $DEF    DATA_T_PHONE      .BLKB  14
        $EQU    DATA_S_PHONE      < . - DATA_T_PHONE>
        $DEF    DATA_T_ADDRESS    .BLKB  30
        $EQU    DATA_S_ADDRESS    < . - DATA_T_ADDRESS>
        $DEF    DATA_T_LOCATION   .BLKB  30
        $EQU    DATA_S_LOCATION   < . - DATA_T_LOCATION>
        $DEF    DATA_L_STATUS     .BLKL  1
        $DEF    DATA_T_SPARE      .BLKB  7
        $EQU    DATA_S_SPARE      < . - DATA_T_SPARE>
        $DEF    DATA_K_LEN
$DEFEND DATA_DEF

$DEFINI MBX_DEF                         ; DECnet command mailbox messages
        $DEF    MBX_MSG           .BLKW  1
        $DEF    MBX_UNIT          .BLKW  1
        $DEF    MBX_NAME_INFO     .BLKB  MAX_NCB
        $DEF    MBX_K_LEN
$DEFEND MBX_DEF

$DEFINI BUFF_DEF                        ; Complete buffer layout
        $DEF    BUFF_L_FLINK      .BLKL  1
        $DEF    BUFF_L_BLINK      .BLKL  1
        $DEF    BUFF_L_ASTID      .BLKL  1
        $DEF    BUFF_Q_IOSB       .BLKQ  1
        $DEF    BUFF_T_DATA_MBX   .BLKB  DATA_K_LEN
        $DEF    BUFF_K_LEN
$DEFEND
        ASSUME  DATA_K_LEN,GE,MBX_K_LEN

$DEFINI ASTID_DEF                       ; I/O completion parameter (ASTPRM)
        $DEF    ASTID_B_TYPE      .BLKB  1
        $DEF    ASTID_B_NDX       .BLKB  1
        $DEF    ASTID_W_UNUSED    .BLKW  1
$DEFEND ASTID_DEF

$DEFINI LCT_DEF                         ; Link control table entry
        $DEF    LCT_W_UNIT        .BLKW  1
        $DEF    LCT_W_CHANNEL     .BLKW  1
        $DEF    LCT_L_CUR_BUFF    .BLKL  1
        $DEF    LCT_K_LEN
$DEFEND LCT_DEF
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
;+
;  Declare external routines and variables
;-

        .EXTRN  LIB$_NOTFOU,-
                LIB$_QUEWASEMP

        .SBTTL  RO_DATA - Read Only DATA
        .PSECT  RO_DATA RD,NOWRT,EXE

NET_DEVICE:     .ASCID  /_NET:/
NETCMD_MBX:     .ASCID  /NETCMD_MBX/

        .SBTTL  RW_DATA - Read Write DATA
        .PSECT  RW_DATA RD,WRT,EXE,QUAD

QUE_HDR:        .BLKQ   2                       ; Queue headers
BUFFERS:        .BLKB   <MAX_BUFFS * BUFF_K_LEN> ; Buffers

DB_FAB:         $FAB    -                       ; Database FAB
                    FNM=<DB_DIR:USER_DB.IDX>,-  ;   File name
                    FAC=<GET>,-                 ;   Get operations
                    MRS=DATA_K_LEN,-            ;   Maximum record
                    ORG=<IDX>,-                 ;   Indexed
                    SHR=<DEL,GET,PUT,UPD>,-     ;   Complete sharing
                    RAT=<CR>                    ;   Carriage return

DB_RAB:         $RAB -                          ; Database RAB
                    FAB=DB_FAB,-                ;   Associated FAB
                    RAC=<KEY>,-                 ;   Key access
                    KRF=0,-                     ;   Key offset
                    KSZ=DATA_S_NAME,-           ;   Key size
                    USZ=DATA_K_LEN              ;   Size of user buffer

OBJECT_NAME:    .ASCID  /DB_SERVER/             ; Network object name

LINK_CONTROL:   .BLKB   <MAX_LINKS * LCT_K_LEN>  ; Link Control Table (LCT)
LCT_ALLOC_MASK: .BLKL   <<<MAX_LINKS-1> / 32> + 1> ; Allocation bit mask

NCB_DESC:       .WORD   MAX_NCB                 ; Descriptor for NCB
                .WORD   0                       ;
                .ADDRESS NCB

NCB:            .BLKB    MAX_NCB                ; Maximum size of NCB
NFB_DESC:       .WORD    NFB_LEN                ; Descriptor for network function block
                .WORD    0
                .ADDRESS NFB

NFB:            .BYTE    NFB$C_DECLNAME         ; Declare name function
                .LONG    0                      ; Terminator
NFB_LEN = < . - NFB>
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
GETDVI_ITM:     .WORD   4                       ; GETDVI item list, length
                .WORD   DVI$_UNIT               ;   Return unit
                .ADDRESS UNIT                   ;   Unit location
                .LONG   0                       ;   Return length
                .LONG   0                       ;   Terminator
IOSB:           .BLKQ   1                       ; I/O status block
CUR_BUFF:       .BLKL   1                       ; Current logical link buffer
NETCMD_BUFF:    .BLKL   1                       ; Current NETCMD buffer
UNIT:           .BLKL   1                       ; Temporary unit location
MBX_CHAN:       .BLKW   1                       ; Channel for NETCMD mailbox
NETDCL_CHAN:    .BLKW   1                       ; _NET: channel
INDEX:          .BLKB   1                       ; LCT index
NET_SHUT:       .BLKB   1                       ; Shutdown flag


        .SBTTL  CODE - Start of program
        .PSECT  CODE    RD,NOWRT,EXE
        .ENTRY  DB_SERVER    ^M<>

;+
;  After initialization, requests are processed until a NETSHUT message
;  is received in the network command mailbox.  ASTs are used to allow
;  asynchronous processing, thus accommodating service of several logical
;  links concurrently.
;-

        BSBW    INITIALIZATION                  ; Initialization
        BLBC    R0,99$                          ; If LBC, error, return status
                                                ; While success and !NET_SHUT
10$:    MOVAL   CUR_BUFF,R2                     ;     Return current buffer
        MOVL    #LIVE_QUE,R1                    ;     Remove from LIVE_QUE
        BSBW    REMQUE_BUFFER                   ;   Remove an entry from the queue
        BLBC    R0,70$                          ;   If LBC, error, check below
        MOVL    CUR_BUFF,R2                     ;   Get buffer address
        MOVAB   BUFF_L_ASTID(R2),R3             ;   Get base of ASTID
        CASEB   ASTID_B_TYPE(R3),-              ; Dispatch the message type
                #NET_RD,#<NET_CMD-NET_RD>
20$:    .WORD   30$-20$                         ;   Network command message
        .WORD   40$-20$                         ;   Logical link read
        .WORD   50$-20$                         ;   Logical link write
        MOVL    #SS$_BADPARAM,R0                ;   Invalid type, set error
        BRB     99$                             ;     and return status
30$:    BSBW    PROCESS_REQUEST                 ;   Process logical link read completion
        BRB     60$                             ;   Check error and shutdown
40$:    BSBW    PROCESS_RESPONSE                ;   Process logical link write completion
        BRB     60$                             ;   Check error and shutdown
50$:    BSBW    PROCESS_NETCMD                  ;   Process DECnet command message
60$:    BLBC    R0,99$                          ;   If LBC, error, return status
        BLBC    NET_SHUT,10$                    ;   If LBC, no shutdown, continue
        BRB     99$                             ;     Shutdown, exit
70$:    CMPL    R0,#LIB$_QUEWASEMP              ;   Queue empty?
        BNEQ    99$                             ;     If NEQ, no, return status
        MOVL    #SS$_NORMAL,R0                  ;     Queue empty, reset status
        $HIBER_S                                ;   Snooze until next request
        BLBC    R0,99$                          ;   If LBC, error, return status
        BRB     10$                             ; Next

99$:    RET
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
INITIALIZATION:
        BSBW    INITIALIZE_VARIABLES            ; Initialize variables and queues
        BLBC    R0,99$                          ; If LBC, error, return status
        BSBW    DECLARE_NETWORK_OBJECT          ; Declare ourself as network object
        BLBC    R0,99$                          ; If LBC, error, return status
        BSBW    OPEN_DATABASE                   ; Open the database
        BLBC    R0,99$                          ; If LBC, error, return status
        BSBW    ISSUE_NETCMD_READ               ; Issue a read on the NETCMD mailbox
99$:    RSB

INITIALIZE_VARIABLES:

        MOVL    #SS$_NORMAL,R0                  ; Start fresh, no errors
        CLRB    NET_SHUT                        ; No NETSHUT yet
        MOVAB   BUFFERS,R2                      ; Get base of buffers
        MOVL    #FREE_QUE,R1                    ; Queue to receive buffers
        MOVAQ   QUE_HDR[R1],R4                  ; Get header address
        CLRL    R3                              ; For R3 = 0 to MAX_BUFFS-1

10$:    INSQTI  (R2),(R4)                       ;   Insert buffer into queue at tail
        ADDL2   #BUFF_K_LEN,R2                  ;   Bump to next buffer
        AOBLSS  #MAX_BUFFS,R3,10$               ; Next R3
        MOVAL   NETCMD_BUFF,R2                  ;   Remove queue buffer
        MOVL    #FREE_QUE,R1                    ;   Remove from FREE_QUE
        BSBW    REMQUE_BUFFER                   ; Remove a buffer from queue
        BLBC    R0,99$                          ; If LBC, error, return status
        MOVL    NETCMD_BUFF,R2                  ; Get buffer for receive
99$:    RSB

DECLARE_NETWORK_OBJECT:
        $CREMBX_S       -                       ; Create mailbox to receive NETCMD messages
                PRMFLG=#TEMP_MBX,-              ;  Temporary mailbox
                CHAN=MBX_CHAN,-                 ;  Channel
                MAXMSG=#MAX_MSG,-               ;  Maximum message size
                BUFQUO=#BUF_QUO,-               ;  Message buffering quota
                LOGNAM=NETCMD_MBX               ;  Mailbox name
        BLBC    R0,99$                          ; If LBC, error, return status
        $ASSIGN_S       -                       ; Assign a channel to _NET:
                DEVNAM=NET_DEVICE,-             ;   _NET: device
                CHAN=NETDCL_CHAN,-              ;   Channel
                MBXNAM=NETCMD_MBX               ;   Associate mailbox for NETCMD messages
        BLBC    R0,99$                          ; If LBC, error, return status
        $QIOW_S         -                       ; Issue network declare function
                CHAN=NETDCL_CHAN,-              ;   Channel to _NET: device
                FUNC=#IO$_ACPCONTROL,-          ;   ACP function
                IOSB=IOSB,-                     ;   I/O status block
                P1=NFB_DESC,-                   ;   Network function block
                P2=#OBJECT_NAME                 ;   Object to declare
        BLBC    R0,99$                          ; If LBC, error, return status
        MOVZWL  IOSB,R0                         ; Get completion status
99$:    RSB
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
OPEN_DATABASE:
        MOVAB   DB_FAB,R6                       ; Set base of FAB
        MOVAB   DB_RAB,R7                       ; Set base of RAB
        $OPEN   (R6)                            ; Open the file
        BLBC    R0,99$                          ; If LBC, error, return status
        $CONNECT (R7)                           ; Connect a stream to file
99$:    RSB

PROCESS_REQUEST:
        BLBS    BUFF_Q_IOSB(R2),10$             ; If LBS, I/O success, process
        MOVZWL  BUFF_Q_IOSB(R2),R0              ; Get the completion status

;+
;  The I/O completed with a failure.  If the status is not a fatal
;  error, treat it as a network failure and recover.  Cleanup will be
;  performed when the formal DECnet command message is delivered.
;-

        BR_FATIO 99$                            ; Branch if fatal status?
        MOVL     #SS$_NORMAL,R0                 ;   Recover status
        MOVL     #FREE_QUE,R1                   ;   Queue to insert buffer
        BSBW     INSQUE_BUFFER                  ; Insert the buffer
        BRB      99$                            ; Return

;+
;  Successful I/O completion
;-
10$:    MOVAB   BUFF_T_DATA_MBX(R2),R3          ; Get address of data
        BSBW    READ_DATABASE                   ; Read the database
        BLBC    R0,99$                          ; If LBC, error, return status
        MOVZBL  BUFF_L_ASTID+ASTID_B_NDX(R2),R4 ; Get the LCT index
        BSBW    ISSUE_LINK_WRITE                ; Return the information
99$:    RSB

PROCESS_RESPONSE:
        BLBS    BUFF_Q_IOSB(R2),10$             ; If LBS, I/O success, process
        MOVZWL  BUFF_Q_IOSB(R2),R0              ; Get the completion status

;+
;  The I/O completed with a failure.  If the status is not a fatal
;  error, treat it as a network failure and recover.  Cleanup will be
;  performed when the formal DECnet command message is delivered.
;-

        BR_FATIO 99$                            ; Branch if fatal I/O status
        MOVL    #SS$_NORMAL,R0                  ;   Recover status
        MOVL    #FREE_QUE,R1                    ;   Queue to insert buffer
        BSBW    INSQUE_BUFFER                   ; Insert the buffer
        BRB     99$                             ; Return

;+
;  Successful I/O completion
;-

10$:    MOVAB   BUFF_T_DATA_MBX(R2),R3          ; Get address of data
        MOVZBL  BUFF_L_ASTID+ASTID_B_NDX(R2),R4 ; Get the LCT index
        BSBW    ISSUE_LINK_READ                 ; Issue a read for next request
99$:    RSB
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
        .ENABL  LSB                             ; Enable LSB for dispatching
PROCESS_NETCMD:
        MOVZWL  BUFF_Q_IOSB(R2),R0              ; Get the I/O completion status
        BLBC    R0,99$                          ; If LBC, error, return status
        MOVAB   BUFF_T_DATA_MBX(R2),R3          ; Get base of data portion
        CASEW   MBX_MSG(R3),#MSG$_ABORT,-       ; Dispatch to appropriate
                #<MSG$_NETSHUT-MSG$_ABORT>      ;   subordinate

10$:    .WORD   ABORT-10$                       ;     ( MSG$_ABORT )
        .WORD   CONFIRM-10$                     ;     ( MSG$_CONFIRM )
        .WORD   CONNECT-10$                     ;     ( MSG$_CONNECT )
        .WORD   DISCON-10$                      ;     ( MSG$_DISCON )
        .WORD   EXIT-10$                        ;     ( MSG$_EXIT )
        .WORD   INTMSG-10$                      ;     ( MSG$_INTMSG )
        .WORD   PATHLOST-10$                    ;     ( MSG$_PATHLOST )
        .WORD   PROTOCOL-10$                    ;     ( MSG$_PROTOCOL )
        .WORD   REJECT-10$                      ;     ( MSG$_REJECT )
        .WORD   THIRDPARTY-10$                  ;     ( MSG$_THIRDPARTY )
        .WORD   TIMEOUT-10$                     ;     ( MSG$_TIMEOUT )
        .WORD   NETSHUT-10$                     ;     ( MSG$_NETSHUT )

        MOVL    #SS$_BADPARAM,R0                ; Unknown message
        BRB     99$                             ; Return status

ABORT:
DISCON:
EXIT:
PATHLOST:
PROTOCOL:
THIRDPARTY:
TIMEOUT:
        MOVZWL  MBX_UNIT(R3),UNIT               ; Link unit number to cleanup
        BSBW    CLEANUP_LINK                    ; Cleanup after failure
        BRB     99$                             ; Return with status

CONFIRM:
INTMSG:
REJECT:
        BSBW    NOT_USED                        ; Messages not used for example
        BRB     99$                             ; Return with status

CONNECT:
        BSBW    ESTABLISH_LINK                  ; Establish link
        BRB     99$                             ; Return with status

NETSHUT:
        BSBW    SHUTDOWN                        ; Shutdown processing
99$:    RSB                                     ; Return with status

        .DSABL   LSB                            ; Enable LSB


CLEANUP_LINK:
        BSBW    FIND_LCT                        ; Locate the LCT entry
        BLBS    R0,10$                          ; If LBS, the entry was found
        MOVL    #SS$_NORMAL,R0                  ; Not found, reset status and
        BRB     20$                             ;   release buffer (timing problem)
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
10$:    MULL3   #LCT_K_LEN,R4,-(SP)             ; Calculate the offset
        MOVAB   LINK_CONTROL,R3                 ; Get base of table
        ADDL2   (SP)+,R3                        ; Add offset to base
        $DASSGN_S    -                          ; Deassign the logical link
                CHAN=LCT_W_CHANNEL(R3)          ;   channel
        BLBC    R0,99$                          ; If LBC, error, return status
        BSBW    RELEASE_LCT                     ; Release the LCT entry
        BLBC    R0,99$                          ; If LBC, error, return status

20$:    MOVL    #FREE_QUE,R1                    ; Insert into FREE_QUE
        BSBW    INSQUE_BUFFER                   ; Insert buffer at tail
99$:    RSB

NOT_USED:

;+
;  Some network command messages are not used in this program.  Insert
;  them into the FREE_QUE and dismiss the event.
;-

    MOVL    #FREE_QUE,R1            ; Insert into FREE_QUE
    BSBW    INSQUE_BUFFER           ; Insert buffer at tail
    RSB

ESTABLISH_LINK:
        ADDL2   #4,R3                           ; Increment past (message/unit)
        MOVZBL  (R3)+,R4                        ; Get byte count of device name
        ADDL2   R4,R3                           ; Skip over device string
        MOVZBL  (R3)+,R4                        ; Get byte count of information
        MOVW    R4,NCB_DESC                     ; Update NCB descriptor (length)
        MOVC3   R4,(R3),NCB                     ; Copy NCB information into NCB
        MOVL    #SS$_NORMAL,R0                  ; Reset status
        MOVL    CUR_BUFF,R2                     ; Reset buffer address
        BSBW    ALLOCATE_LCT                    ; Allocate a link control entry
        BLBS    R0,10$                          ; If LSB, entry available

;+
;  Reject the connection, no LCT entries are available.
;-
        $QIOW_S -                               ; Reject the request
                CHAN=NETDCL_CHAN,-              ;   _NET: channel
                FUNC=#IO$_ACCESS!IO$M_ABORT,-   ;   Abort the request
                IOSB=IOSB,-                     ;   I/O status block
                P2=#NCB_DESC                    ;   NCB of request
        BLBC    R0,05$                          ; If LBC, error, return status
        MOVL    #FREE_QUE,R1                    ;   Insert into FREE_QUE
        BSBW    INSQUE_BUFFER                   ; Insert buffer at tail
05$:    BRW     99$                             ; Return with status

10$:    MOVAB   LINK_CONTROL,R3                 ; Get base of LCT
        MULL3   #LCT_K_LEN,R4,-(SP)             ; Calculate offset for index
        ADDL2   (SP)+,R3                        ; Calculate address of LCT entry
        $ASSIGN_S       -                       ; Assign a channel for logical link
                DEVNAM=NET_DEVICE,-             ;   _NET: device
                CHAN=LCT_W_CHANNEL(R3),-        ;   Channel
                MBXNAM=NETCMD_MBX               ;   DECnet command mailbox
        BLBC    R0,05$                          ; If LBC, error, return status
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
        $GETDVIW_S      -                       ; Request unit of logical link channel
                CHAN=LCT_W_CHANNEL(R3),-        ;   Logical link channel
                ITMLST=GETDVI_ITM,-             ;   Item list (unit)
                IOSB=IOSB                       ;   I/O status block
        BLBC    R0,99$                          ; If LBC, error, return status
        MOVZWL  IOSB,R0                         ; Get completion status
        BLBC    R0,99$                          ; If LBC, error, return status
        MOVW    UNIT,LCT_W_UNIT(R3)             ; Insert unit into table

        $QIOW_S -                               ; Accept the connect initiate request
                CHAN=LCT_W_CHANNEL(R3),-        ;   Logical link channel
                FUNC=#IO$_ACCESS,-              ;   Accept
                IOSB=IOSB,-                     ;   I/O status block
                P2=#NCB_DESC                    ;   Network Connect Block (NCB)
        BLBC    R0,99$                          ; If LBC, error, return status
        MOVZWL  IOSB,R0                         ; Get completion status
        BLBS    R0,20$                          ; If LBS, success, continue

;+
;  Check completion status and recover from network errors.
;-

        BR_FATACC 99$                           ; Branch if fatal accept error?
        MOVL    #SS$_NORMAL,R0                  ;   Recover network error
        BSBW    CLEANUP_LINK                    ; Cleanup the link
        BRB     99$                             ;   and return status

20$:    MOVL    R2,LCT_L_CUR_BUFF(R3)           ; Insert current buffer
        BSBW    ISSUE_LINK_READ                 ; Issue a read on link
99$:    RSB

SHUTDOWN:

;+
;  A NETSHUT message was received.  Set a flag to drop through the main
;  processing loop and terminate.
;-

        MOVB    #1,NET_SHUT                     ; Indicate shutdown
        MOVL    #FREE_QUE,R1                    ;   Insert into FREE_QUE
        BSBW    INSQUE_BUFFER                   ; Insert buffer at tail
        RSB

ISSUE_NETCMD_READ:
        MOVL    #NET_CMD,BUFF_L_ASTID(R2)       ; Set AST type
        $QIO_S  -                               ; Issue asynchronous read
                CHAN=MBX_CHAN,-                 ;   Network command mailbox
                FUNC=#IO$_READVBLK,-            ;   Read request
                IOSB=BUFF_Q_IOSB(R2),-          ;   I/O status block
                ASTADR=AST_ROUTINE,-            ;   AST routine
                ASTPRM=BUFF_L_ASTID(R2),-       ;   AST parameter
                P1=BUFF_T_DATA_MBX(R2),-        ;   Buffer area
                P2=#MBX_K_LEN                   ;   Maximum size for receive
        RSB                                     ; Return with status

ISSUE_LINK_READ:
        MOVAB   BUFF_L_ASTID(R2),R3             ; Get base of ASTID
        MOVB    R4,ASTID_B_NDX(R3)              ; Set index of LCT
        MOVB    #NET_RD,ASTID_B_TYPE(R3)        ; Get I/O type
        MULL3   #LCT_K_LEN,R4,-(SP)             ; Calculate offset for index
        MOVAB   LINK_CONTROL,R3                 ; Get base of table
        ADDL2   (SP)+,R3                        ; Calculate address of entry
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
        $QIO_S  -                                   ; Issue asynchronous read on logical link
                CHAN=LCT_W_CHANNEL(R3),-            ;   Logical link channel
                FUNC=#IO$_READVBLK,-                ;   Read operation
                IOSB=BUFF_Q_IOSB(R2),-              ;   I/O status block
                ASTADR=AST_ROUTINE,-                ;   AST routine
                ASTPRM=BUFF_L_ASTID(R2),-           ;   AST parameter
                P1=BUFF_T_DATA_MBX(R2),-            ;   Data area
                P2=#DATA_K_LEN                      ;   Maximum size of read
        RSB

ISSUE_LINK_WRITE:
        MOVAB   BUFF_L_ASTID(R2),R3                 ; Get base of ASTID
        MOVB    R4,ASTID_B_NDX(R3)                  ; Set index of LCT
        MOVB    #NET_WRT,ASTID_B_TYPE(R3)           ; Get I/O type
        MULL3   #LCT_K_LEN,R4,-(SP)                 ; Calculate offset for index
        MOVAB   LINK_CONTROL,R3                     ; Get base of table
        ADDL2   (SP)+,R3                            ; Calculate address of entry
        $QIO_S  -                                   ; Issue asynchronous read on logical link
                CHAN=LCT_W_CHANNEL(R3),-            ;   Logical link channel
                FUNC=#IO$_WRITEVBLK,-               ;   Write operation
                IOSB=BUFF_Q_IOSB(R2),-              ;   I/O status block
                ASTADR=AST_ROUTINE,-                ;   AST routine
                ASTPRM=BUFF_L_ASTID(R2),-           ;   AST parameter
                P1=BUFF_T_DATA_MBX(R2),-            ;   Data area
                P2=#DATA_K_LEN                      ;   Size of write
        RSB

READ_DATABASE:
        MOVAB   DB_RAB,R4                           ; Get base of RAB
        MOVAB   DATA_T_NAME(R3),RAB$L_KBF(R4)       ; Key address
        MOVL    R3,RAB$L_UBF(R4)                    ; User buffer
        $GET    (R4)                                ; Read the record
        MOVL    R0,DATA_L_STATUS(R3)                ; Return status to user
        BLBS    R0,99$                              ; If LBS, the record was found
        CMPL    #RMS$_RNF,R0                        ; Recoverable error?
        BNEQ    99$                                 ;   If NEQ, nonrecoverable
        MOVL    #SS$_NORMAL,R0                      ;   Recoverable reset status
99$:    RSB

FIND_LCT:
        MOVAB   LINK_CONTROL,R3                     ; Get base of table
        CLRL    R4                                  ; While not found and < MAX_LINKS
10$:    BBC     R4,LCT_ALLOC_MASK,20$               ;   If BC, entry NOT in use
        CMPW    LCT_W_UNIT(R3),UNIT                 ;     In use, entry match?
        BEQL    99$                                 ;       If EQL, found it
20$:    ADDL2   #LCT_K_LEN,R3                       ;       No match, continue
        AOBLSS  #MAX_LINKS,R4,10$                   ; Next
        MOVL    #LIB$_NOTFOU,R0                     ; Not found, return status
99$:    RSB

ALLOCATE_LCT:
        FFC     #0,#MAX_LINKS,LCT_ALLOC_MASK,R4     ; Available entry?
        BNEQ    10$                                 ;   If NEQ, available entry
        MOVL    #LIB$_NOTFOU,R0                     ;   Table full, set and
        BRB     99$                                 ;      return status
10$:    BBSS    R4,LCT_ALLOC_MASK,99$               ; Mark entry in use
99$:    RSB
```

Example 8–4 Cont'd. on next page

Example 8–4 (Cont.) Nontransparent Communication Using System Services

```
RELEASE_LCT:
        BBCC    R4,LCT_ALLOC_MASK,10$           ; Clear allocated flag
10$:    MULL3   R4,#LCT_K_LEN,-(SP)             ; Calculate offset for index
        MOVAB   LINK_CONTROL,R1                 ; Get base of LCT
        ADDL2   (SP)+,R1                        ; Calculate address of entry
        CLRW    LCT_W_UNIT(R1)                  ; Clear unit
        CLRW    LCT_W_CHANNEL(R1)               ; Clear channel
        CLRL    LCT_L_CUR_BUFF(R1)              ; Clear current buffer
        RSB
REMQUE_BUFFER:
        MOVAQ   QUE_HDR[R1],R3                  ; Get address of header
        REMQHI  (R3),(R2)                       ; Remove a buffer
        BVC     99$                             ; If VC, an entry was removed
        MOVL    #LIB$_QUEWASEMP,R0              ; Queue was empty, return status
99$:    RSB

INSQUE_BUFFER:
        MOVAQ   QUE_HDR[R1],R3                  ; Get address of header
        INSQTI  (R2),(R3)                       ; Insert the buffer
        RSB
        .ENABL   LSB
        .ENTRY   AST_ROUTINE    ^M<R2,R3>
        MOVL     #SS$_NORMAL,R0                 ; Cleanup trash from AST delivery
        MOVZBL   4+ASTID_B_TYPE(AP),R1          ; Get the ASTID
        MOVZBL   4+ASTID_B_NDX(AP),R3           ; Get the LCT index
        CASEB    R1,#NET_RD,#<NET_CMD-NET_RD>   ; Dispatch to
10$:    .WORD    QUE_BUFF-10$                   ;   NET_RD, insert in queue in live
        .WORD    QUE_BUFF-10$                   ;   NET_WRT, insert in queue in live
        .WORD    QUE_AND_REISSUE-10$            ;   NET_CMD, reissue
        MOVL     #SS$_BADPARAM,R0               ; Invalid value
        BRB      99$

QUE_BUFF:
        MOVAB    LINK_CONTROL,R2                ; Get base of table
        MULL3    #LCT_K_LEN,R3,-(SP)            ; Calculate offset for index
        ADDL2    (SP)+,R2                       ; Add offset to base
        MOVL     LCT_L_CUR_BUFF(R2),R2          ;   Buffer to insert
        MOVL    #LIVE_QUE,R1                    ;   Insert into live
        BSBW    INSQUE_BUFFER                   ; Insert buffer at tail
        BRB     90$                             ; Return

QUE_AND_REISSUE:
        MOVL    NETCMD_BUFF,R2                  ;   Buffer to insert
        MOVL    #LIVE_QUE,R1                    ;   Insert into live
        BSBW    INSQUE_BUFFER                   ; Insert buffer at tail
        MOVAL   NETCMD_BUFF,R2                  ;   Address for buffer
        MOVL    #FREE_QUE,R1                    ;   Remove from free
        BSBW    REMQUE_BUFFER                   ; Remove buffer from head
        BLBC    R0,99$                          ; If LBC, error, terminate
        MOVL    NETCMD_BUFF,R2                  ; Set buffer in which to read
        BSBW    ISSUE_NETCMD_READ               ; Issue another read
        BLBC    R0,99$                          ; If LBC, error, terminate
90$:    $WAKE_S                                 ; Wake the main process
        BLBC    R0,99$                          ; If LBC, error, terminate
        RET
99$:    $EXIT_S R0                              ; Exit with status
        .DSABL  LSB
        .END    DB_SERVER
```

9 File Operations in a Heterogeneous Network Environment

This chapter contains material to assist you in using DECnet–VAX Version 5.0 to initiate remote file operations in a heterogeneous network environment. This chapter discusses restrictions on using DCL commands and RMS service calls to access files on the following types of remote systems:

VMS to IAS
VMS to P/OS
VMS to RSTS/E
VMS to RSX using RMS-based FAL
VMS to RSX using FCS-based FAL
VMS to RT–11
VMS to TOPS–10
VMS to TOPS–20
VMS to MS–DOS
VMS to Ultrix
VMS to MVS
VMS Version 5.0 to previous VMS version

The chapter is organized by operating-system type: one section for each heterogeneous system with which your VMS operating system running DECnet–VAX Version 5.0 may communicate. Each section describes differences in file system operation between the two systems and constraints on the use of VMS file processing commands. The restrictions on the remote file operations you can perform from a VMS Version 5.0 node to a particular heterogeneous node result from file system design differences and DECnet implementation restrictions between the systems.

Specifically, the appropriate section for each remote system itemizes the VMS Record Management Services (RMS) features that are supported between DECnet–VAX Version 5.0 systems, but are not supported when accessing files on the heterogeneous system. The chapter also discusses limitations on the DIGITAL Command Language (DCL) commands that you can use when communicating with the remote node. Throughout this chapter comments are provided to help you handle the differences in file system design.

The most recent version of DECnet used by each heterogeneous system is represented in this chapter.

9.1 General DECnet–VAX Restrictions

This section is a brief summary of VMS RMS features that are not supported by DECnet–VAX for remote file access. The list is not complete; it is meant only to highlight the more important differences between local and remote file access capabilities. For more complete information on this subject, refer to the description of the various RMS control blocks in the *VMS Record Management Services Manual*.

9.1 General DECnet–VAX Restrictions

- The following VMS RMS service calls are not supported for network use:

  $ENTER $NXTVOL $REMOVE

- The Terminal XAB is not supported for network operations; it is ignored.
- Protection XAB fields that support access control lists are ignored for network operations.
- Only one data stream per open file is allowed. That is, the multistream (MSE) bit option of the file sharing (SHR) field of the FAB is not supported for network use.
- Access to files on magnetic tapes mounted on a remote VMS operating system is not supported. You can, however, copy files from a local magnetic tape to disk on a remote node.
- When multiple Allocation XABs are linked to the FAB, they must be in ascending order by area number (AID field). Similarly, when multiple Key Definition XABs are used, they must be in ascending order by key of reference (REF field).
- File protection information may not be completely preserved if the two nodes do not fully support each other's protection attributes. An example of this incompatibility occurs between RSX-11M/M-PLUS and VMS operating systems. Although both systems represent their protection masks as RWED, RSX-11M/M-PLUS interprets that as Read, Write, Extend, and Delete, while the VMS operating system interprets RWED as Read, Write, Execute, and Delete. This results in the "E" protection field being unmappable between these two systems.
- The Journaling XABs are not supported.
- File monitoring is not supported.

9.2 VMS to IAS Network Operation

This section pertains to a VMS node communicating with an IAS node running DECnet–IAS Version 3.0. The discussion focuses on file operations initiated from the VMS node to access remote files by means of the FAL at the IAS node.

The restrictions described in the following subsections are related to incompatible features in file system design between the two operating systems.

9.2.1 File Formats and Access Modes

The following types of file and access method are not supported by the VMS operating system when communicating with an IAS node:

- File organizations and record formats

Sequential	Stream (STM)
	Stream_CR (STMCR)
	Stream_LF (STMLF)
	Variable with fixed control (VFC) where fixed header size is not 2 bytes
Relative	All formats
Indexed	All formats

- Record attributes

  Print file carriage control (PRN)

- File access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address
  Block I/O

You can copy a sequential file in VFC format from a VMS node to an IAS node, provided the file has a 2-byte fixed header with a carriage control attribute other than print file. To transfer a file that has print file carriage control, such as a VMS batch log file, enter the following command:

```
CONVERT/FDL=VAR.FDL input-file output-file
```

The FDL control file VAR.FDL contains the following information:

```
FILE
        ORGANIZATION            sequential
RECORD
        FORMAT                  variable
        CARRIAGE_CONTROL        carriage_return
```

The CONVERT command and associated FDL control file transforms the input file to variable-length format with implied carriage control and copies it to the remote node according to the output file specification.

9.2.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and an IAS node:

- VMS RMS service calls

$DELETE	$DISPLAY	$EXTEND	$FIND
$FREE	$READ	$RELEASE	$RENAME
$REWIND	$SPACE	$TRUNCATE	$UPDATE
$WRITE			

- RMS extended attribute blocks

  Allocation XAB
  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  CBT (contiguous-best-try) bit of FOP field
  DEQ (default extend quantity) field

9.2.3 File Specifications

The general format of a file specification for naming a file on a remote IAS system is as follows:

```
node::device:[directory]name.type;version
```

The following are major differences in syntax between file specifications used on IAS and on VMS:

- IAS does not support dollar sign ($), underscore (_) and hyphen (-) characters in file name components.
- IAS does not recognize the percent sign (%) as a valid wildcard character.
- The directory component of an IAS file specification cannot be a named directory list, such as [A.B.C]; it must be in UIC (user identification code) format, such as [100,3].
- The file name component has a maximum length of nine characters and the file type cannot exceed three characters. IAS systems return an error if you specify a longer file name or file type.
- IAS uses octal version numbers in file specifications whereas VMS uses decimal version numbers.

9.2.4 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and an IAS node:

- ANALYZE/RMS_FILE
- BACKUP
- OPEN/WRITE
- RENAME

9.2.4.1 APPEND

Using the APPEND command, you are limited to appending one local input file to the output file residing on the IAS node.

9.2.4.2 COPY

The /EXTENSION and /PROTECTION qualifiers for the COPY command are not supported and are ignored if specified.

File creation date and time information is not preserved on a file copy operation to an IAS node where wildcards are used in the output file specification. Instead, the current date and time are used as the file creation date and time.

Because the IAS operating system uses octal version numbers in file specifications, an attempt to copy a file with a version number containing an 8 or 9 is rejected by the remote system, as shown in the following example:

```
$ COPY A.DAT;9 IAS::*.*
%COPY-E-OPENOUT, error opening _IAS::A.DAT;9 as output
-RMS-F-FNM, error in file name
```

There are two ways to circumvent this problem. You can either specify an appropriate octal version number in the output file specification, or you can specify a null or zero version number in the output file specification to force highest version number processing on the remote node. This latter technique is particularly useful when several files are copied with one DCL command. For example:

```
$ COPY A.DAT;9  IAS::A.DAT;11
$ COPY B.DAT;28 IAS::*.*;
$ COPY B.DAT;28 IAS::*.*;0
$ COPY *.DAT    IAS::*.*;0
```

9.3 VMS to P/OS Network Operation

This section pertains to a VMS node communicating with a P/OS node running DECnet–PRO V2.0. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the P/OS node.

The following restrictions are related to incompatible features in file system design between the two operating systems.

9.3.1 File Formats and Access Modes

The following types of file and record attribute are not supported by VMS when communicating with a P/OS node:

- File organizations and record formats

Sequential	Stream_CR (STMCR)
	Stream_LF (STMLF)
Indexed	All prologue 3 formats
	With 64-bit binary (BN8) key types
	With 64-bit integer (IN8) key types
	With collating (COL) key types
	With descending key types (DSTG, DIN2, DBN2, DIN4, DBN4, DIN8, DBN8, DPAC, DCOL)

- Record attributes

  Record attributes are compatible.

- File access modes

  Modes are compatible.

9.3.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and a P/OS node:

- VMS RMS service call

  $RELEASE

- Significant fields and bit options of the FAB

  CBT (contiguous-best-try) bit of FOP field
  SCF (submit command file) bit of FOP field
  SPL (spool file) bit of FOP field

9.3.3 File Specifications

The general format of a file specification for naming a file on a remote P/OS system is as follows:

node::device:[directory]name.type;version

The following are major differences in syntax between file specifications used on P/OS and VMS:

- P/OS does not support dollar sign ($), underscore (_) and hyphen (-) characters in file name components.
- The directory component in a P/OS file specification cannot be a named directory list, such as [A.B.C]; it can be a single directory name, such as [USERFILES], or it can be expressed in UIC (user identification code) format, such as [15,1].
- The file name component has a maximum length of nine characters and the file type cannot exceed three characters. P/OS systems return an error if you specify a longer file name or file type.

9.3.4 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and a P/OS node:

- OPEN/WRITE
- PRINT/REMOTE
- SUBMIT/REMOTE

9.4 VMS to RSTS/E Network Operation

This section pertains to a VMS node communicating with a RSTS/E node running DECnet/E Version 3.0. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the RSTS/E node.

The following restrictions are related to incompatible features in file system design between the two systems.

9.4.1 File Formats and Access Modes

The following types of file and access method are not supported by VMS when communicating with a RSTS/E node:

- File organizations and record formats

Sequential	Stream_CR (STMCR)
	Stream_LF (STMLF)
Indexed	All prologue 3 formats
	With 64-bit binary (BN8) key types
	With 64-bit integer (IN8) key types
	With collating (COL) key types
	With descending key types (DSTG, DIN2, DBN2, DIN4, DBN4, DIN8, DBN8, DPAC, DCOL)

- Record attributes

  Attributes are compatible.

- File access modes

  Random access by key value
  Random access by record file address

DECnet/E does not support record mode access to indexed files; it supports only block I/O access to indexed files.

Note that an attempt to access an indexed file located on a RSTS/E node in record mode results in an RMS-F-BUG_DAP error instead of an RMS-F-SUPPORT error.

9.4.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and a RSTS/E node:

- VMS RMS service calls

$DISPLAY	$EXTEND	$FREE	$RELEASE
$RENAME	$SPACE	$TRUNCATE	

- RMS extended attribute blocks

  Allocation XAB
  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  CBT (contiguous-best-try) bit of FOP field
  DEQ (default extend quantity) field

9.4.3 File Specifications

The general format of a file specification for naming a file on a remote RSTS/E operating system is as follows:

```
node::device:[directory]name.type
```

The following are major differences in syntax between file specifications used on RSTS/E and on VMS:

- RSTS/E does not support dollar sign ($), underscore (_) and hyphen (-) characters in file name components, except for the special use of the dollar sign at the start of a file name.
- RSTS/E does not recognize the percent sign (%) as a valid wildcard character.
- The directory component of a RSTS/E file specification cannot be a named directory list, such as [A.B.C]; it must be in UIC (user identification code) format, such as [1,2]. RSTS/E operating systems, however, express UICs in decimal radix, whereas VMS operating systems use octal numbers. On the RSTS/E operating system, the UIC is referred to as a PPN (project programmer number).

  To access a RSTS/E file whose directory component in PPN format contains decimal digits, use the quoted string form of the file specification. For example:

  ```
  $ TYPE RSTS::"SY:[9,18]TEST.DAT"
  ```

- The file name component has a maximum length of six characters and the file type cannot exceed three characters. If you specify a longer file name, RSTS/E truncates the name to six characters.
- RSTS/E does not support version numbers. It accepts a file specification containing a version number without returning an error, but ignores the version number.

9.4.4 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and a RSTS/E node:

- PURGE
- RENAME

9.4.4.1 APPEND

In using the APPEND command, you are limited to appending one local input file to an output file on the RSTS/E node.

9.4.4.2 COPY

The /EXTENSION and /PROTECTION qualifiers for the COPY command are not supported and are ignored if specified.

File creation date and time information is not preserved on a file copy operation to a RSTS/E node where wildcards are used in the output file specification. Instead, the current date and time are used as the file creation date and time.

Because RSTS/E does not support version numbers in file specifications (it ignores any version number supplied), an attempt to copy a file with an explicit version number fails if a file with the same name and type already exists at the RSTS/E node. For example, if a file with the name RSTS::TEST.DAT already exists on the remote node, an attempt to update it by copying a new version of that file to the node produces the following results:

```
$ COPY TEST.DAT;2 RSTS::*.*
%COPY-E-OPENOUT, error opening _RSTS::TEST.DAT;2 as output
-RMS-E-FEX, file already exists, not superseded
```

9.4.4.3 DELETE

If you use the DELETE command with a wildcard file specification to delete several files from a directory on a remote RSTS/E node, the operation may appear to complete successfully even though some of the files may remain in the directory. This behavior is caused by a file system incompatibility in the way VMS and RSTS/E perform wildcard file deletion operations. This problem occurs only if the remote directory has at least 30 files cataloged.

To determine if all the files you specify have been deleted successfully, enter a DIRECTORY command to examine the remote directory. Then repeat the wildcard DELETE command if necessary to remove unwanted files. If the number of files you are attempting to delete is small, using the /LOG qualifier with the DELETE command may help you to determine if all the files have been deleted.

9.4.4.4 DIRECTORY

When you enter a DIRECTORY/FULL command to examine a RSTS/E file, the information displayed differs from that displayed for a VMS file, in the following respects:

- The file owner is displayed as [0,0] if the owner of the file is identified by a UIC that contains decimal digits.
- The file REVISION number shown is either 0 or 1. A REVISION number of 0 indicates the file has not been revised; a REVISION number of 1 indicates the file has been revised.

- Under the attributes of an indexed file, information about the number of keys, the number of areas, and the prologue version number of the file is not displayed. This information is omitted because the RSTS/E FAL does not return file attribute information stored in the prologue portion of an indexed file.
- Under the attributes of a relative file, the maximum record number is displayed as 0.

9.4.4.5 DUMP/RECORDS and TYPE Commands

Because RSTS/E does not support record mode access (nonblock I/O access) to indexed files, you cannot use the DCL commands DUMP/RECORDS and TYPE to examine indexed files located on the remote RSTS/E node.

9.5 VMS to RSX Network Operation Using RMS-Based FAL

This section pertains to a VMS node communicating with an RSX node running either DECnet–11M Version 4.0 or DECnet–11M–PLUS Version 2.0 where the RSX File Access Listener (FAL) calls RMS–11 to perform local file operations. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the RSX node.

The following restrictions are related to incompatible features in file system design between the two systems.

9.5.1 File Formats and Access Modes

The following types of file and record attributes are not supported by VMS when communicating with an RSX node running the RMS-based FAL:

- File organizations and record formats

Sequential	Stream_CR (STMCR)
	Stream_LF (STMLF)
Indexed	All prologue 3 formats
	With 64-bit binary (BN8) key types
	With 64-bit integer (IN8) key types
	With collating (COL) key types
	With descending key types (DSTG, DIN2, DBN2, DIN4, DBN4, DIN8, DBN8, DPAC, DCOL)

- Record attributes

  Record attributes are compatible.

- File access modes

  Modes are compatible.

9.5.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and an RMS-based RSX node:

- VMS RMS service call

  $RELEASE

- Significant fields and bit options of the FAB

  CBT (contiguous-best-try) bit of FOP

9.5.3 File Specifications

The general format of a file specification for naming a file on a remote RSX–11M or RSX–11M–PLUS system is as follows:

node::device:[directory]name.type;version

The following are major differences in syntax between file specifications used on RSX and VMS:

- RSX operating systems do not support dollar sign ($), underscore (_) and hyphen (-) characters in file name components.
- The directory component in an RSX file specification cannot be a named directory list, such as [A.B.C]; it must be in UIC (user identification code) format, such as [15,1].
- The file name component has a maximum length of nine characters and the file type cannot exceed three characters. RSX operating systems return an error if you specify a longer file name or file type.
- RSX operating systems use octal version numbers in file specifications whereas the VMS operating system uses decimal version numbers.

9.5.4 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following is not supported between VMS and an RMS-based RSX node:

- OPEN/WRITE

9.5.4.1 COPY

Because RSX–11M and RSX–11M–PLUS operating systems use octal version numbers in file specifications, an attempt to copy a file with a version number containing an 8 or 9 is rejected by the remote system. For example:

```
$ COPY A.DAT;9 RSX::*.*
%COPY-E-OPENOUT, error opening RSX::A.DAT;9 as output
-RMS-F-FNM, error in file name
```

There are two ways to circumvent this problem. You can specify an appropriate octal version number in the output file specification, or you can specify a null or zero version number in the output file specification to force highest version number processing on the remote node. This latter technique is particularly useful when several files are copied with one DCL command. For example:

```
$ COPY A.DAT;9  RSX::A.DAT;11
$ COPY B.DAT;28 RSX::*.*;
$ COPY B.DAT;28 RSX::*.*;0
$ COPY *.DAT    RSX::*.*;0
```

9.6 VMS to RSX Network Operation Using FCS-Based FAL

This section pertains to a VMS node communicating with an RSX node running DECnet–11M Version 4.0 or DECnet–11M–PLUS V2.0 where the RSX FAL calls the File Control Services (FCS–11) to perform file operations. The discussion focuses on file operations initiated from the VMS node to access remote files by means of the FAL at the RSX node.

The following restrictions are related to incompatible features in file system design between the two systems.

9.6.1 File Formats and Access Modes

The following types of file and access method are not supported by VMS when communicating with an RSX node running the FCS-based FAL:

- File organizations and record formats

Sequential	Stream (STM)
	Stream_CR (STMCR)
	Stream_LF (STMLF)
	Variable with fixed control (VFC) where fixed header size is not 2 bytes
Relative	All formats
Indexed	All formats

- Record attributes

  Print file carriage control (PRN)

- File access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address
  Block I/O

You can copy a sequential file in VFC format from a VMS node to an FCS-based RSX node, provided the file has a 2-byte fixed header with a carriage control attribute other than print file. To transfer a file that has print file carriage control, such as a VMS batch log file, enter the following command:

```
$ CONVERT/FDL=VAR.FDL input-file output-file
```

The FDL control file VAR.FDL contains the following information:

```
FILE
        ORGANIZATION            sequential
RECORD
        FORMAT                  variable
        CARRIAGE_CONTROL        carriage_return
```

The CONVERT command and associated FDL control file transform the input file to variable-length format with implied carriage control and then copy it to the remote node according to the output file specification.

9.6.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and an FCS-based RSX node:

- VMS RMS service calls

  $DELETE $DISPLAY $EXTEND $FIND
  $FREE $READ $RELEASE $RENAME
  $REWIND $SPACE $TRUNCATE $UPDATE
  $WRITE

- RMS extended attribute blocks

  Allocation XAB
  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  CBT (contiguous-best-try) bit of FOP field
  DEQ (default extension quantity) field

9.6.3 File Specifications

The general format of a file specification for naming a file on a remote RSX-11M or RSX-11M-PLUS system is as follows:

node::device:[directory]name.type;version

The following are major differences in syntax between file specifications used on RSX and on VMS:

- RSX operating systems do not support dollar sign ($), underscore (_) and hyphen (-) characters in file name components.
- The directory component in an RSX file specification cannot be a named directory list, such as [A.B.C]; it must be in UIC (user identification code) format, such as [15,1].
- The file name component has a maximum length of nine characters and the file type cannot exceed three characters. RSX operating systems return an error if you specify a longer file name or file type.
- RSX operating systems use octal version numbers in file specifications whereas VMS uses decimal version numbers.

9.6.4 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and an FCS-based RSX node:

- ANALYZE/RMS_FILE
- BACKUP
- OPEN/WRITE
- RENAME

9.6.4.1 APPEND

In using the APPEND command, you are limited to appending one local input file to an output file residing on the FCS-based RSX node.

9.6.4.2 COPY

The /EXTENSION and /PROTECTION qualifiers for the COPY command are not supported and are ignored if specified.

File creation date and time information is not preserved on a file copy operation to an RSX node where wildcards are used in the output file specification. Instead, the current date and time are used as the file creation date and time.

Because RSX-11M and RSX-11M-PLUS operating systems use octal version numbers in file specifications, an attempt to copy a file with a version number containing an 8 or 9 is rejected by the remote system, as follows:

```
$ COPY A.DAT;9 RSX::*.*
%COPY-E-OPENOUT, error opening RSX::A.DAT;9 as output
-RMS-F-FNM, error in file name
```

There are two ways to circumvent this problem. You can either specify an appropriate octal version number in the output file specification, or you can specify a null or zero version number in the output file specification to force highest version number processing on the remote node. This latter technique is particularly useful when several files are copied with one DCL command. For example:

```
$ COPY A.DAT;9  RSX::A.DAT;11
$ COPY B.DAT;28 RSX::*.*;
$ COPY B.DAT;28 RSX::*.*;0
$ COPY *.DAT    RSX::*.*;0
```

9.7 VMS to RT-11 Network Operations

This section pertains to a VMS node communicating with an RT-11 node running DECnet-RT Version 2.1. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the RT-11 node.

9.7.1 File System Constraints

The file systems used by RT–11 and VMS are dissimilar in many respects. A fundamental difference between them involves the handling of file attribute information. When you create a file on a VMS operating system, attribute information about the file is stored in a header block on disk for use when the file is subsequently opened. The implication is that the structure of an established file cannot change. In contrast, RT–11 does not save attribute information such as file format with a file; it expects you to provide this information when you open the file. File attribute information, however, is not an input to VMS RMS when you open a file.

To provide transparent access to files on a remote RT–11 operating system, VMS RMS restricts the types of file that you can create and open on the remote node. When you access an RT–11 file in record mode, VMS RMS treats the file as having stream format. Block I/O access is permitted; the remote file is viewed as having fixed length 512 byte records where virtual block number is translated to relative record number.

9.7.1.1 File Formats and Access Modes

The following types of file and access method are not supported by VMS when communicating with an RT–11 node:

- File organizations and record formats

Sequential	Fixed length (FIX) without implied carriage control
	Stream_CR (STMCR)
	Stream_LF (STMLF)
	Variable length (VAR) without implied carriage control
	Variable with fixed control (VFC)
Relative	All formats
Indexed	All formats

- Record attributes

  FORTRAN carriage control (FTN)
  Print file carriage control (PRN)
  None specified (embedded carriage control)

- Record access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address

For record mode access, the only file type in common between the two systems is a sequential file in STM (stream) format. For convenience, however, when you are transferring a file to an RT–11 node, VMS RMS automatically converts a VMS sequential file with fixed or variable format and implied carriage control to a sequential file with stream format and embedded carriage control. This automatic conversion is performed during a file create operation, and VMS RMS returns an alternate success code (RMS$_CVT_STM) to indicate that the file format has been modified.

Note also that, when a stream format file is retrieved from an RT–11 node, VAX RMS automatically changes the record attribute from embedded carriage control to implied carriage control.

In general, you can copy text files created by the SOS Editor without line numbers being saved or by the EDT Editor to an RT–11 operating system. VMS batch log files and files created by the SOS Editor with line numbers intact, however, are stored in VFC format and cannot be copied to an RT–11 system in that form. To transfer this type of file, enter the following DCL command:

```
$ CONVERT/FDL=STM.FDL input-file output-file
```

The FDL control file STM.FDL contains the following information:

```
FILE
        ORGANIZATION            sequential
RECORD
        FORMAT                  stream
        CARRIAGE_CONTROL        none
```

The CONVERT command and associated FDL control file transform the input file to stream format with embedded carriage control and copies it to the remote node according to the output file specification.

9.7.1.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and an RT–11 node:

- VMS RMS service calls

  $DELETE $DISPLAY $EXTEND $FIND
  $FREE $RELEASE $RENAME $REWIND
  $SPACE $TRUNCATE $UPDATE

- RMS extended attribute blocks

  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  ALQ (allocation quantity) field
  DEQ (default extend quantity) field
  CBT (contiguous-best-try) bit of FOP field
  CTG (contiguous) bit of FOP field
  SCF (submit command file) bit of FOP field
  SPL (spool file) bit of FOP field

- Significant fields and bit options of the RAB

  EOF (position to end of file) bit of ROP field

9.7.2 File Specifications

The general format of a file specification for naming a file on a remote RT–11 operating system is as follows:

node::device:name.type

The following are major differences in syntax between file specifications on RT–11 and VMS:

- RT–11 does not support dollar sign ($), underscore (_) and hyphen (-) characters in file name components.
- RT–11 does not recognize the percent sign (%) as a valid wildcard character.
- RT–11 does not have a directory component ir its file specification.
- The file name component has a maximum length of six characters and the file type cannot exceed three characters. If you specify a longer file name or file type, RT–11 returns an error.
- RT–11 does not support version numbers. Specification of a version number, however, is permitted when you refer to an RT–11 file, because VMS RMS discards any version number before sending the file specified to the RT–11 FAL.

9.7.3 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and an RT–11 node:

- ANALYZE/RMS_FILE
- APPEND
- BACKUP
- OPEN/WRITE
- PRINT/REMOTE
- PURGE
- RENAME
- SUBMIT/REMOTE

9.7.3.1 COPY

The /ALLOCATION, /CONTIGUOUS, /EXTENSION, and /PROTECTION qualifiers for the COPY command are not supported and are ignored if specified.

Using COPY to merge several files into a single output file is not supported.

RT–11 does not support version numbers in file specifications and supersedes files by default. Therefore, if you attempt to copy a file with the same name and type as one that already exists on the remote RT–11 node, the new file supersedes the old one. No warning message is displayed.

9.7.3.2 DELETE

The DCL command DELETE requires that you specify an explicit or wildcard version number in the file specification. However, because RT–11 does not accept a file specification containing a version number, VMS RMS removes the version number before sending the file specification to the RT–11 operating system. To satisfy the requirements of both systems, specify a null version number in the file specification, as follows:

```
DELETE RT::TEST.DAT;
```

9.8 VMS to TOPS–10 Network Operations

This section pertains to a VMS node communicating with a TOPS–10 node running DECnet–10 Version 4.0. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the TOPS–10 node.

9.8.1 File System Constraints

The file systems used by TOPS–10 and VMS are dissimilar in many respects. A fundamental difference between them involves the handling of file attribute information. When you create a file on a VMS operating system, attribute information about the file is stored in a header block on disk for use when the file is subsequently opened. The implication is that the structure of an established file cannot change. In contrast, TOPS–10 does not save attribute information such as file format with a file; it expects you to provide this information when you open the file. File attribute information, however, is not an input to VMS RMS when you open a file.

To provide transparent access to files on a remote TOPS–10 system, VMS RMS restricts the types of file that you can create and open on the remote node. When you access a TOPS–10 file in record mode, VMS RMS treats the file as having stream format.

9.8.1.1 File Formats and Access Modes

Because of differences in file system design, the following types of file and access method are not supported by VMS when communicating with a TOPS–10 node:

- File organizations and record formats

Sequential	Fixed length (FIX) without implied carriage control
	Stream_CR (STMCR)
	Stream_LF (STMLF)
	Variable length (VAR) without implied carriage control
	Variable with fixed control (VFC)
Relative	All formats
Indexed	All formats

- Record attributes

  FORTRAN carriage control (FTN)
  Print file carriage control (PRN)
  None specified (embedded carriage control)

- Record access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address
  Block I/O

For record mode access, the only file type in common between the two systems is a sequential file in STM (stream) format. For convenience, however, when you are transferring a file to a TOPS–10 node, VMS RMS automatically converts a VMS sequential file with fixed or variable format and implied carriage control to a sequential file with stream format and embedded carriage control. This automatic conversion is performed during a file create operation, and VMS RMS returns an alternate success code (RMS$_CVT_STM) to indicate that the file format has been modified.

Note also that when a stream format file is retrieved from a TOPS–10 node, VMS RMS automatically changes the record attribute from embedded carriage control to implied carriage control.

In general, you can copy text files created by the TPU or the EDT Editor to a TOPS–10 operating system. VMS batch log files, however, are stored in VFC format, and cannot be copied in that form to a TOPS–10 operating system. To transfer this type of file, enter the following DCL command:

```
$ CONVERT/FDL=STM.FDL input-file output-file
```

The FDL control file STM.FDL contains the following information:

```
FILE
        ORGANIZATION            sequential
RECORD
        FORMAT                  stream
        CARRIAGE_CONTROL        none
```

The CONVERT command and associated FDL control file transform the input file to stream format with embedded carriage control and then copy them to the remote node according to the output file specification.

9.8.1.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and a TOPS–10 node:

- VMS RMS service calls

  $DELETE $DISPLAY $EXTEND $FIND
  $FREE $READ $RELEASE $RENAME
  $REWIND $SPACE $TRUNCATE $UPDATE
  $WRITE

- RMS extended attribute blocks

  Allocation XAB
  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  ALQ (allocation quantity) field
  DEQ (default extend quantity) field
  CBT (contiguous-best-try) bit of FOP field

9.8.1.3 File Specifications

The general format of a file specification for naming a file on a remote TOPS–10 operating system is as follows:

```
node::device:[directory]name.type
```

The following are the major differences in syntax between file specifications on TOPS–10 and on VMS:

- The directory component of a TOPS–10 file specification is in PPN (project programmer number) format, such as [3655,7031], where the two numbers are in octal radix. The directory component can also be in extended PPN format containing up to five levels of subdirectories. An example of a directory component in extended PPN format is [10,20,A,B,C,D,E].

  The VMS operating system cannot parse directory components in PPN format (with numbers larger than 377 octal) or handle extended PPN formats containing subdirectories. The DECnet–10 implementation, however, does accept directory components using period (.) instead of comma (,) delimiters, and converts commas to periods when returning file specifications to VMS operating systems. Consequently, when you enter a file specification for a remote TOPS–10 operating system, use the VMS named directory list format for expressing TOPS–10 PPNs and extended PPNs. For example, use [3655.7031] or [10.20.A.B.C.D.E] to specify a directory component.

- The file name component has a maximum length of six characters and the file type cannot exceed three characters. If you specify a longer file name, TOPS–10 truncates the name to six characters.

- TOPS–10 does not support version numbers. It accepts a file specification containing a version number without returning an error, but ignores the version number.

9.8.2 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and a TOPS–10 node:

- ANALYZE/RMS_FILE
- APPEND
- BACKUP
- OPEN/WRITE
- RENAME

9.8.2.1 COPY

The /ALLOCATION and /EXTENSION qualifiers to the COPY command are not supported and are ignored if specified.

9.8.2.2 DIRECTORY

When you enter a DIRECTORY/FULL command to examine a TOPS–10 file, the information displayed differs in the following respects from that displayed for a VMS file:

- The file owner is displayed as [0,0] to indicate that this information is not available.
- The file REVISION number is not shown and file REVISION date and time information is not available from the TOPS–10 operating system.
- The blocks used and blocks allocated values displayed, which indicate the size of the file, refer to 128-word pages (providing 640 bytes of storage), not 512-byte blocks.

9.9 VMS to TOPS–20 Network Operations

This section pertains to a VMS node communicating with a TOPS–20 node running DECnet–20 Version 3.0. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the TOPS–20 node.

9.9.1 File System Constraints

The file systems used by TOPS–20 and VMS are dissimilar in many respects. A fundamental difference between them involves the handling of file attribute information. When you create a file on a VMS operating system, attribute information about the file is stored in a header block on disk for use when the file is subsequently opened. The implication is that the structure of an established file cannot change. In contrast, TOPS–20 does not save attribute information such as file format with a file; it expects you to provide this information when you open the file. File attribute information, however, is not an input to VMS RMS when a file is opened.

To provide transparent access to files on a remote TOPS–20 operating system, VMS RMS restricts the types of file that you can create and open on the remote node. When you access a TOPS–20 file in record mode, VMS RMS treats the file as having stream format. Although block I/O is supported by DECnet–20, it is not supported between VMS and TOPS–20 because the block sizes are different.

9.9.1.1 File Formats and Access Modes

Because of differences in file system design, the following types of file and access method are not supported by VMS when communicating with a TOPS–20 node:

- File organizations and record formats

Sequential	Fixed length (FIX) without implied carriage control
	Stream_CR (STMCR)
	Stream_LF (STMLF)
	Variable length (VAR) without implied carriage control
	Variable with fixed control (VFC)
Relative	All formats
Indexed	All formats

- Record attributes

  FORTRAN carriage control (FTN)
  Print file carriage control (PRN)
  None specified (embedded carriage control)

- Record access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address
  Block I/O

For record mode access, the only file type in common between the two systems is a sequential file in STM (stream) format. For convenience, however, when you are transferring a file to a TOPS–20 node, VMS RMS automatically converts a VMS sequential file with fixed or variable format and implied carriage control to a sequential file with stream format and embedded carriage control. This automatic conversion is performed during a file create operation, and VMS RMS returns an alternate success code (RMS$_CVT_STM) to indicate that the file format has been modified.

Note also that when a stream format file is retrieved from a TOPS–20 node, VMS RMS automatically changes the record attribute from embedded carriage control to implied carriage control.

In general, you can copy text files created by the TPU or the EDT Editor to a TOPS–20 operating system. VMS batch log files, however, are stored in VFC format, and cannot be copied in that form to a TOPS–20 operating system. To transfer this type of file, enter the following DCL command:

```
$ CONVERT/FDL=STM.FDL input-file output-file
```

The FDL control file STM.FDL contains the following information:

```
FILE
        ORGANIZATION            sequential
RECORD
        FORMAT                  stream
        CARRIAGE_CONTROL        none
```

The CONVERT command and associated FDL control file transform the input file to stream format with embedded carriage control and then copy it to the remote node according to the output file specification.

9.9.1.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and a TOPS–20 node:

- VMS RMS service calls

$DELETE	$DISPLAY	$EXTEND	$FIND
$FREE	$READ	$RELEASE	$RENAME
$REWIND	$SPACE	$TRUNCATE	$UPDATE
$WRITE			

- RMS extended attribute blocks

  Allocation XAB
  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  ALQ (allocation quantity) field
  DEQ (default extend quantity) field
  CBT (contiguous-best-try) bit of FOP field
  CTG (contiguous) bit of FOP field

- Significant fields and bit options of the RAB

  EOF (position to end of file) bit of ROP field

9.9.1.3 File Specifications

The general format of a file specification for naming a file on a remote TOPS–20 system is as follows

```
node::device <directory> name.type.version
```

The following are the major differences in syntax between file specifications on TOPS–20 and on VMS:

- TOPS–20 uses angle brackets (<>) to delimit the directory string instead of square brackets ([]). To facilitate communication with TOPS–20, VMS RMS recognizes angle brackets as valid directory component delimiters.
- TOPS–20 uses the period (.) to delimit the version number instead of the semicolon (;). However, you can specify either a period or a semicolon because VMS RMS converts a semicolon version number delimiter to a period before sending the file specification to the TOPS–20 FAL.

9.9.2 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and a TOPS–20 node:

- ANALYZE/RMS_FILE
- APPEND
- BACKUP
- OPEN/WRITE
- RENAME

9.9.2.1 COPY

The /ALLOCATION, /CONTIGUOUS, /EXTENSION, and /PROTECTION qualifiers to the COPY command are not supported and are ignored if specified.

File creation date and time are not preserved during a file copy operation.

Using COPY to merge several files into a single output file is not supported.

9.9.2.2 DIRECTORY

When you use a DIRECTORY/FULL command to examine a TOPS–20 file, the information displayed differs in the following respects from that displayed for a VMS file:

- The file owner is displayed as [0,0] to indicate that this information is not available.
- The file REVISION number is not shown.
- The blocks used and blocks allocated values displayed, which indicate the size of the file, refer to 128-word pages (providing 640 bytes of storage), not 512-byte blocks.
- TOPS–20 does not have the equivalent of world protection, so this attribute is displayed as a null string.

9.10 VMS to MS–DOS Network Operations

This section pertains to a VMS node communicating with an MS–DOS node running DECnet–DOS Version 1.2, DECnet–Rainbow Version 1.2, or DECnet–VAXmate Version 1.2. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the MS–DOS node.

9.10.1 File System Constraints

The file systems used by MS–DOS and VMS are dissimilar in many respects. A fundamental difference between them involves the handling of file attribute information. When you create a file on a VMS operating system, attribute information about the file is stored in a header block on disk for use when the file is subsequently opened. The implication is that the structure of an established file cannot change. In contrast, MS–DOS does not save attribute information such as file format with a file; it expects you to provide this information when you open the file. File attribute information, however, is not an input to VMS RMS when a file is opened.

To provide transparent access to files on a remote MS–DOS system, VMS RMS restricts the types of file that you can create and open on the remote node. When you access an MS–DOS file in record mode, VMS RMS treats the file as having stream format.

9.10.1.1 File Formats and Access Modes

Because of differences in file system design, the following types of file and access method are not supported by VMS when communicating with an MS–DOS node:

- File organizations and record formats

Sequential	Fixed length (FIX) without implied carriage control
	Stream_CR (STMCR)
	Stream_LF (STMLF)
	Variable length (VAR) without implied carriage control
	Variable with fixed control (VFC)
Relative	All formats
Indexed	All formats

- Record attributes

  FORTRAN carriage control (FTN)
  Print file carriage control (PRN)
  None specified (embedded carriage control)

- Record access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address

For record mode access, the only file type in common between the two systems is a sequential file in STM (stream) format. For convenience, however, when you are transferring a file to an MS–DOS node, VMS RMS automatically converts a VMS sequential file with fixed or variable format and implied carriage control to a sequential file with stream format and embedded carriage control. This automatic conversion is performed during a file create operation, and VMS RMS returns an alternate success code (RMS$_CVT_STM) to indicate that the file format has been modified.

Note also that when a stream format file is retrieved from an MS–DOS node, VMS RMS automatically changes the record attribute from embedded carriage control to implied carriage control.

In general, you can copy text files created by the TPU or EDT Editor to an MS–DOS operating system. VMS batch log files, however, are stored in VFC format, and cannot be copied in that form to an MS–DOS operating system. To transfer this type of file, enter the following DCL command:

```
$ CONVERT/FDL=STM.FDL input-file output-file
```

The FDL control file STM.FDL contains the following information:

```
FILE
        ORGANIZATION            sequential
RECORD
        FORMAT                  stream
        CARRIAGE_CONTROL        none
```

The CONVERT command and associated FDL control file transform the input file to stream format with embedded carriage control and then copy it to the remote node according to the output file specification.

9.10.1.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and an MS–DOS node:

- VMS RMS service calls

  $DELETE $DISPLAY $EXTEND $FIND
  $FREE $RELEASE $RENAME $REWIND
  $TRUNCATE $UPDATE $WRITE

- RMS extended attribute blocks

  Allocation XAB
  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  ALQ (allocation quantity) field
  DEQ (default extend quantity) field
  CBT (contiguous-best-try) bit of FOP field

9.10.1.3 File Specifications

The general format of a file specification for naming a file on a remote MS–DOS operating system is as follows:

```
node::"device:\directory\name"
```

The major difference in syntax between file specifications on MS–DOS and on VMS is that the directory components of an MS–DOS file specification are in an incompatible format. For example:

```
\directory\
```

As a result, you must use quoted strings when you access these MS–DOS files from a VMS operating system.

On DECnet–RB/DOS/VM Version 1.2 systems, the FAL object accepts incoming requests using file specifications in VMS syntax and maps those requests to file specifications for DOS. For example:

```
$ DIRECTORY PC::[REPORT]
```

This directory specification is mapped to the following directory specification:

```
$ DIRECTORY PC::\report\*.*
```

DOS file specifications are restricted to file names of eight characters, file extensions of three characters, and do not support version numbers.

9.10.2 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and an MS–DOS node:

- ANALYZE/RMS_FILE
- APPEND
- BACKUP
- OPEN/WRITE
- RENAME

9.10.2.1 COPY

The /ALLOCATION and /EXTENSION qualifiers to the COPY command are not supported and are ignored if specified.

9.10.2.2 DIRECTORY

When you enter a DIRECTORY/FULL command to examine an MS–DOS file, the information displayed differs in the following respects from that displayed for a VMS file:

- The file owner identifier is displayed as [0,0] to indicate that this information is not available.
- The file ID identifier is displayed as NONE to indicate that this information is not available.
- The file attributes version limit identifier is displayed as 0 to indicate that this information is not available.
- The file REVISION number is not shown and file REVISION date and time information is not available from the MS–DOS operating system.

9.11 VMS to Ultrix Network Operations

This section pertains to a VMS node communicating with an Ultrix node running DECnet–Ultrix Version 1.0. The discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL at the Ultrix node.

9.11.1 File System Constraints

The file systems used by Ultrix and VMS are dissimilar in many respects. A fundamental difference between them involves the handling of file attribute information. When you create a file on a VMS operating system, attribute information about the file is stored in a header block on disk for use when the file is subsequently opened. The implication is that the structure of an established file cannot change. In contrast, Ultrix does not save attribute information such as file format with a file; it expects you to provide this information when you open the file. File attribute information, however, is not an input to VMS RMS when a file is opened.

To provide transparent access to files on a remote Ultrix operating system, VMS RMS restricts the types of file that you can create and open on the remote node. When you access an Ultrix file in record mode, VMS RMS treats the file as having STREAM_LF (STMLF) format.

9.11.1.1 File Formats and Access Modes

Because of differences in file system design, the following types of file and access method are not supported by VMS when communicating with an Ultrix node:

- File organizations and record formats

Sequential	Fixed length (FIX) without implied carriage control
	Stream_CR (STMCR)
	Stream (STM)
	Variable length (VAR) without implied carriage control
	Variable with fixed control (VFC)
Relative	All formats
Indexed	All formats

- Record attributes

  FORTRAN carriage control (FTN)
  Print file carriage control (PRN)
  None specified (embedded carriage control)

- Record access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address
  Block I/O

For record mode access, the only file type in common between the two systems is a sequential file in STMLF (STREAM_LF) format. For convenience, however, when you are transferring a file to an Ultrix node, VMS RMS automatically converts a VMS sequential file with fixed or variable format and implied carriage control to a sequential file with stream format and embedded carriage control. This automatic conversion is performed during a file create operation, and VMS RMS returns an alternate success code (RMS$_CVT_STM) to indicate that the file format has been modified.

Note also that when a STREAM-LF format file is retrieved from an Ultrix node, VMS RMS automatically changes the record attribute from embedded carriage control to implied carriage control.

To transfer files that cannot be directly copied, enter the following DCL command:

```
$ CONVERT/FDL=STMLF.FDL input-file output-file
```

The FDL control file STMLF.FDL contains the following information:

```
FILE
        ORGANIZATION            sequential
RECORD
        FORMAT                  Stream_LF
        CARRIAGE_CONTROL        none
```

The CONVERT command and associated FDL control file transform the input file to stream format with embedded carriage control and then copy it to the remote node according to the output file specification.

9.11.1.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and an Ultrix node:

- VMS RMS service calls

  $DELETE $DISPLAY $EXTEND $FIND
  $FREE $RELEASE $RENAME $REWIND
  $TRUNCATE $UPDATE

- RMS extended attribute blocks

  Allocation XAB
  Key Definition XAB
  Summary XAB

- Significant fields and bit options of the FAB

  ALQ (allocation quantity) field
  DEQ (default extend quantity) field
  CBT (contiguous-best-try) bit of FOP field

9.11.1.3 File Specifications

The general format of a file specification for naming a file on a remote Ultrix operating system is as follows:

node::name

The following are the major differences in syntax between file specifications on Ultrix and on VMS:

- No explicit device names are allowed. Instead, Ultrix has a concept of special files.
- File names on Ultrix are case sensitive (uppercase or lowercase).

Because of these differences, most accesses to an Ultrix operating system require a foreign file specification. Without the foreign file specification syntax, the name is converted to uppercase by VMS, and is then unlikely to match files on the Ultrix operating system. The VMS concepts of device and directory do not match the Ultrix concept of path, nor does Ultrix support separate file type or version fields. Therefore, VMS-related name processing does not work with Ultrix file names.

9.11.2 DCL Considerations

Of the VMS DCL commands that you can use over the network, the following are not supported between VMS and an Ultrix node:

- ANALYZE/RMS_FILE
- BACKUP
- OPEN/WRITE
- RENAME

9.11.2.1 COPY

The /ALLOCATION and /EXTENSION qualifiers to the COPY command are not supported and are ignored if specified.

9.11.2.2 DIRECTORY

When you enter a DIRECTORY/FULL command to examine an Ultrix file, the information displayed differs in the following respects from that displayed for a VMS file:

- The file owner is displayed as [0,0] to indicate that this information is not available.
- The file REVISION number is not shown and file REVISION date and time information is not available from the Ultrix operating system.

9.12 VMS to MVS Network Operations

This section pertains to a VMS node communicating with an IBM MVS operating system. In order to perform file operations, the MVS and VMS operating systems must have the following DIGITAL products installed:

- DECnet/SNA VMS Data Transfer Facility Client (VMS/DTF) on the VMS node

  or

- DECnet/SNA VMS Data Transfer Facility Server (VMS/DTF) on the VMS node

  and

- DECnet/SNA MVS Data Transfer Facility (MVS/DTF) on the MVS node

In addition, your DECnet network must contain a DECnet/SNA Gateway node or a VMS node running the VMS/SNA product.

The following discussion focuses on file operations initiated from the VMS node, to access remote files by means of the FAL on the MVS operating system. The FAL is part of the MVS/DTF product.

The following sections provide a general overview about which file operations are possible and which are not. For a more detailed discussion, refer to the VMS/DTF and MVS/DTF documentation sets.

9.12.1 File System Constraints

The DECnet/SNA Data Transfer Facility (DTF) software makes MVS datasets appear to the VMS operating system as remote RMS files that you can access using RMS calls or utilities (such as COPY) that are layered upon RMS. The underlying differences in the file systems used by MVS and VMS impose a number of constraints on accessing MVS datasets. (Note that files on an IBM operating system are called datasets.)

9.12.1.1 File Formats and Access Modes

Because of differences in file system design, the following types of file and access method are not supported by VMS when communicating with an MVS operating system:

- File organization and record format

Sequential	Stream (STM)
	Stream_CR (STMCR)
	Stream_LF (STMLF)
	Undefined (UDF)
	Variable with fixed control (VFC). When creating a dataset on the MVS operating system, you may specify VFC format if you also specify the record attribute PRINT CARRIAGE_CONTROL. When this dataset is subsequently opened by RMS, it has record format VARIABLE and a record attribute of CARRIAGE_RETURN CARRIAGE_CONTROL. If this dataset is copied back to a VMS operating system, the resultant VMS file has similar attributes; that is, the FAB$C_VFC FAB$V_PRN options are transformed to FAB$C_VAR and FAB$V_CR.
Relative	All formats
Indexed	All formats

- Record attributes

  No carriage control. You must specify either FABV_CR, FABV_FTN or FAB$V_PRN when creating a dataset on the MVS operating system.

- Record access modes

  Random access by relative record number
  Random access by key value
  Random access by record file address
  Block I/O

MVS sequential files that reside on disk or tape are created using the following access methods:

- BSAM (Basic Sequential Access Method)
- QSAM (Queued Sequential Access Method)

These MVS sequential files appear to VMS as RMS sequential files. Partitioned Dataset (PDS) members also appear to VMS as RMS sequential files. Datasets created using the VSAM access method are not supported by Version 1.0 of the DECnet/SNA Data Transfer Facility.

Files that you cannot copy to or from the IBM operating system using the DCL COPY command, because of the previously mentioned constraints, can be copied using the DCL CONVERT command and a suitable FDL control file.

The CONVERT command and associated FDL control file transform the input file to a format supported by the remote MVS operating system by the DTF software.

For record mode access, the only file organization in common between the two systems is a sequential file.

9.12.1.2 VMS RMS Interface

The following VMS RMS features, supported between two VMS nodes, are not supported between a VMS node and an MVS node:

- VMS RMS service calls

$DELETE	$ENTER	$EXTEND	$FIND
$FLUSH	$FREE	$NXTVOL	$READ
$RELEASE	$REMOVE	$RENAME	$REWIND
$SPACE	$TRUNCATE	$UPDATE	$WRITE

- RMS extended attribute blocks

  Key Definition XAB
  Protection XAB
  Revision Date and Time XAB
  Summary XAB

9.12.1.3 File Specifications

The general format of a file specification for naming a dataset on the remote MVS operating system is as follows:

```
DTF-server-node"SNADTF"::"aaa.bbb.ccc.../qual1:val1/qual2:val2..."
```

or

```
DTF-server-node"SNADTF"::"aaa.bbb.ccc...(ddd)/qual1:val1/qual2:val2..."
```

9.12.2 DCL Considerations

Most of the VMS DCL file manipulation commands that can be used over the network can be used to access datasets on an MVS operating system. Any commands that use RMS features, detailed previously as unsupported, do not work, for example:

- BACKUP
- LIBRARIAN
- LINK
- RENAME

9.13 VMS to VMS Network Operations (Version 5.0 to Previous Version)

This section pertains to file operations initiated on a VMS Version 5.0 node running DECnet–VAX Version 5.0 where the remote system is a VMS node running a previous release of DECnet–VAX.

The following restriction indicates a new feature not previously supported by DECnet–VAX.

The following type of file is not supported by VMS when communicating with a VMS node running a previous DECnet–VAX release:

- File organization and record format

Indexed	With collating (COL) key type
	With descending collating (DCOL) key type

A Area Routing Configuration

Phase IV DECnet supports area routing, which permits the configuration of networks in which the nodes are grouped into areas. This appendix presents recommendations and guidelines for configuring networks that use area routing. It illustrates the guidelines with an example of the design of a multiple-area network, and indicates the NCP commands required to build the configuration database for this network. This appendix also recommends a procedure for converting an existing network to a multiple-area network. Section A.5 describes problems that can occur when you are configuring an area-based network, and includes suggestions for solving these problems. Section A.6 discusses area routing on the Ethernet.

Area routing concepts are described in detail in Section 2.4. Area routing techniques enable configuration of a network consisting of a number of areas; each area is a group of nodes that forms a subnetwork. DECnet supports routing of packets within areas and a second level of routing between areas. The router that performs routing within an area is called a level 1 router; the router that performs routing to and from other areas as well as within its own area is called a level 2 router (or area router).

Each level 1 router keeps information on the state of all nodes in its area, but not on the state of nodes outside its area. It routes all packets addressed to nodes outside its area to the nearest level 2 router. Each level 2 router keeps information on the least-cost path to areas throughout the network, as well as the state of the nodes in its own area. When a level 1 router receives a packet destined for a node in another area, it uses level 1 routing to send the packet to the nearest level 2 router in its own area. The level 2 router forwards the packet along the least-cost path to the nearest level 2 router outside its area. The packet is transmitted along a level 2 path to the level 2 router in the destination area; this level 2 router sends the packet by level 1 routers to the destination node.

Thus, a basic reason for dividing a network into multiple areas is to reduce the amount of routing traffic that occurs in a single-area network.

A.1 Area Routing Configuration Guidelines

Configuration of a network that consists of multiple areas is more complex than configuration of a network that, by default, consists of a single area. The design of a multiple-area network introduces a second, higher level of routing that links the areas. Designing a network for area routing involves awareness of certain network topological restrictions unique to area routing configurations. The following area routing configuration guidelines are based on these restrictions. The guidelines are intended to prevent problems such as loss of routing path, isolation of nodes, or incorrect routing of packets. These potential problems are discussed in Section A.5.

Area Routing Configuration

A.1 Area Routing Configuration Guidelines

When you configure a multiple-area network, you should follow these guidelines:

- Each node must belong in only one area. This restriction applies to all nodes in the network, Phase III nodes as well as Phase IV nodes. Phase III nodes must be logically associated with a single area even though they are not assigned an area number by network management, and must not have circuits outside the area.
- Only a level 2 router can establish a circuit with a node in another area, thus enabling communication between the areas. A level 1 router cannot have any circuits outside its own area.
- Within a network, the level 2 routers must form a subnetwork; that is, they must be connected in such a way that they create a network of their own. There must be a level 2 routing path between any pair of level 2 routers across the network. Level 1 routers do not forward level 2 routing information.
- Treat each area as though it were a separate network. Each area must be physically intact and capable of running on its own. Within the area, there must be a level 2 path between any pair of level 2 routers.
- Provide enough redundancy within each area and between areas to avoid having a single point of failure in the network. For redundancy within an area, you could include more than one level 2 router and provide for alternate paths between nodes. This redundancy prevents loss of the routing path within the area or isolation of any one node. For redundancy between areas, you could provide for alternate paths between areas so that loss of a line does not disconnect any area from the rest of the network. Complete redundancy may not be feasible, however, for small networks.
- Place all Phase III nodes on the periphery in each area. Do not place a Phase III node in a path between two Phase IV nodes. A Phase III node cannot communicate directly with nodes in other areas or with nodes in the same area that have addresses greater than 255.
- Do not link a Phase III node in one area with a node in another area. Such a connection could lead to area leakage, a problem described in Section A.5.2.2.

The recommended approach to designing a multiple-area network configuration is to begin by designing the level 2 routers, area by area, into a level 2 subnetwork. Then, in each area, add the level 1 routers. Finally, add the end nodes required to complete each area. This design approach is illustrated in the following section.

A.2 Designing a Multiple-Area Network

This section demonstrates the use of the configuration guidelines for designing a multiple-area network. The goal of the design process is to build a robust, redundant network that is not subject to a single point of failure.

Figure A–1 shows the level 2 routers as a subnetwork of a multiple-area network. For purposes of illustration, DMR lines are used to connect level 2 routers within each area and DMC lines to connect level 2 routers in different areas. In each area, the level 2 routers are configured in pairs for redundancy and connected by enough DMR lines so that the loss of one DMR line does not prevent the flow of level 2 routing traffic through the area. Redundancy between different areas is achieved by the way in which the DMC lines connect level 2 routers in the different areas. If one of the DMC lines fails, level 2 routing traffic can still reach each area by an alternate path. In area 9, the redundant level 2 routers that form part of a VAXcluster are connected by a CI line.

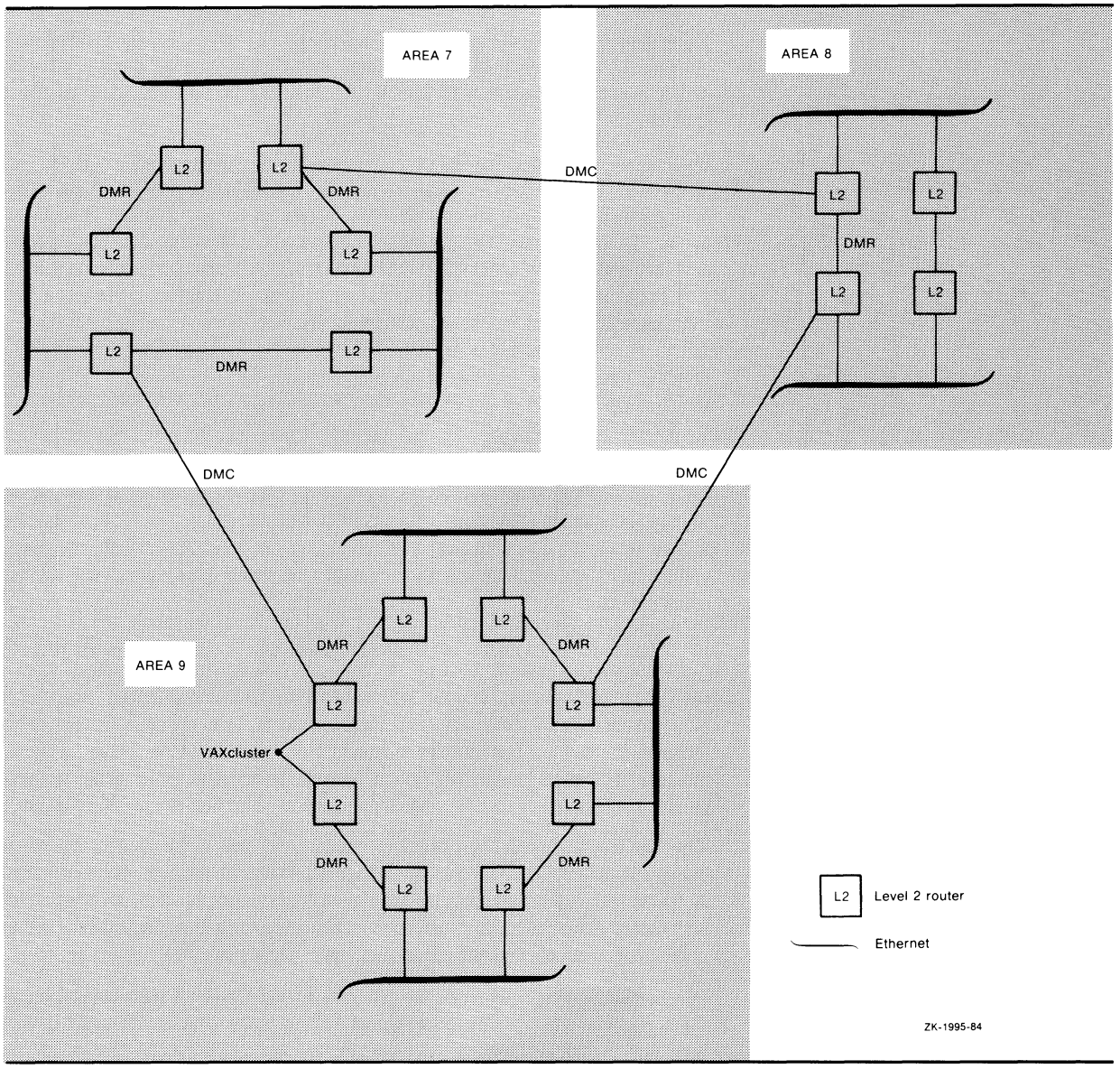
Figure A–2 shows the next stage of the design process: adding the level 1 routers and the end nodes to each area. The figure does not include all the nodes that may be required to make the network complete. It illustrates only a few typical uses of level 1 routers and end nodes, indicating the way in which you could add such nodes to the level 2 subnetwork to complete the network design.

In Figure A–2, in area 7, an end node and a level 1 router are attached directly to the first Ethernet. VMS end nodes are connected to the level 1 router by means of DDCMP asynchronous lines. Another end node is connected to a level 2 router attached to the second Ethernet (lower left) in area 7. Because this end node is not on a routing path between level 2 routers, it could possibly be a Phase III end node. In area 9, two level 1 routers are added to the redundant level 2 routers in a VAXcluster.

The network design shown in Figure A–2 ensures a robust network not vulnerable to a line or node failure that could isolate or bring down an area. A fully redundant multiple-area network, such as that in Figure A–2, may not be practical for smaller networks, however. Redundancy is a desirable design goal, not a requirement, in a multiple-area network.

When you complete the design of a multiple-area network, begin network configuration by configuring each area separately, as though it were a network by itself. The following section shows an example of the NCP commands required to configure area 7 of the network in Figure A–2. After you configure all areas in the network and they are running and stable, connect the areas.

Figure A–1 Level 2 Router Subnetwork of a Multiple-Area Network

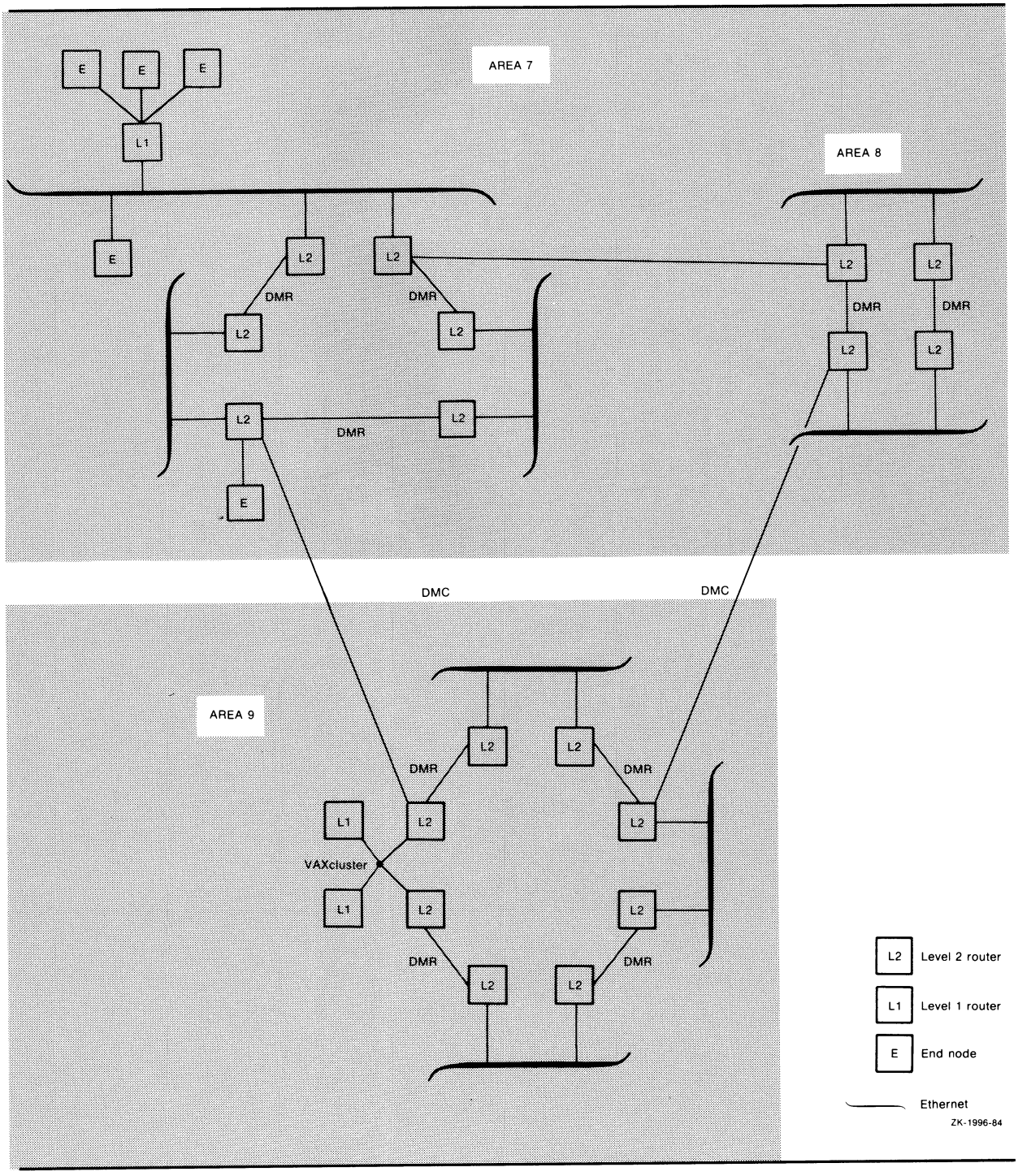


A.3 Sample Multiple-Area Network Configuration

The example in this section illustrates how you can use NCP commands to configure a multiple-area network. It lists the NCP commands required to build the permanent database for one area of the large multiple-area network shown in Figure A–2, and indicates how to complete the network configuration.

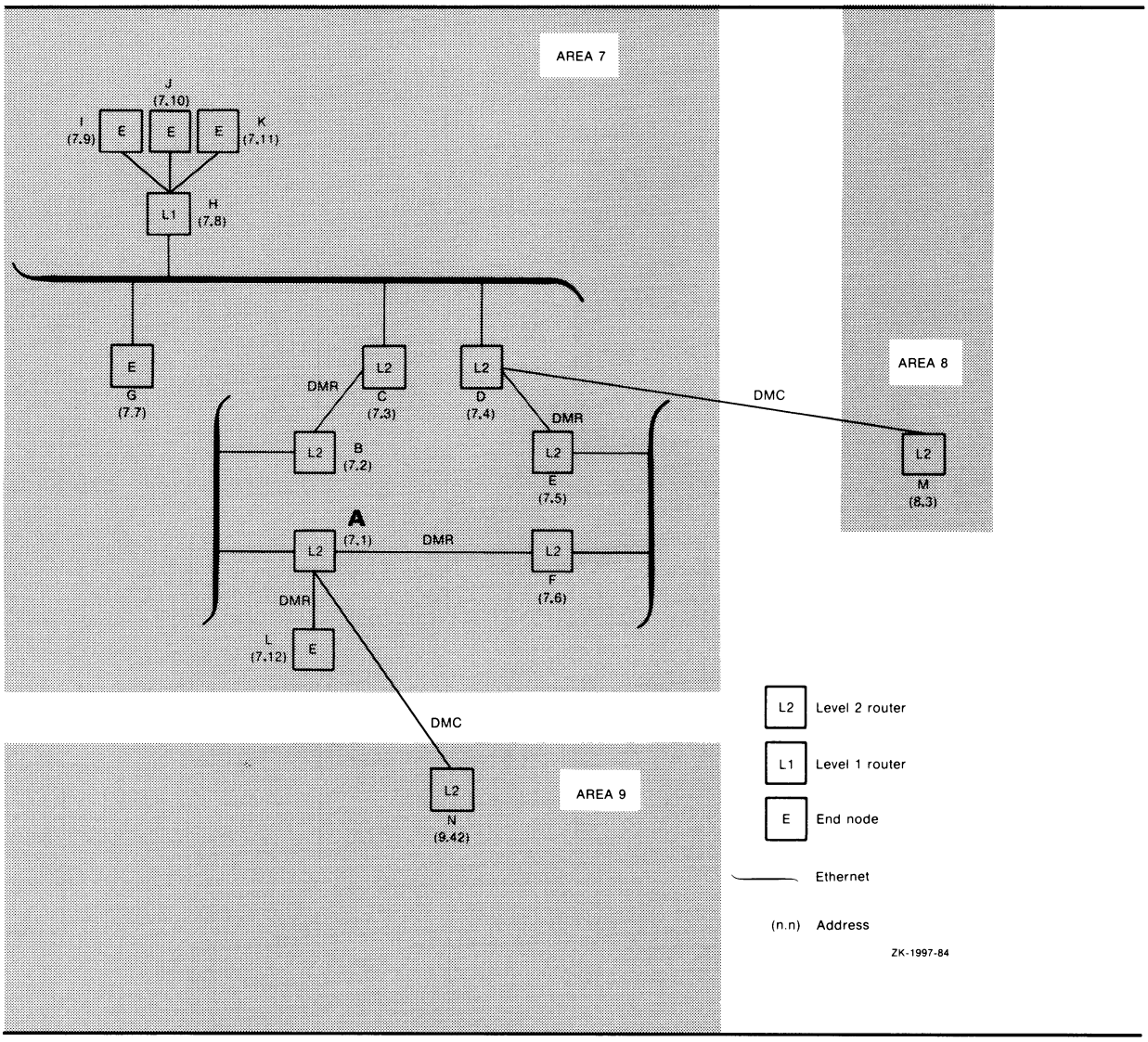
This example builds a database for a network configuration of twelve nodes in area 7, as depicted in Figure A–3. Area 7 is connected to areas 8 and 9. The database being built is for node A. You would enter similar commands to create the database for each of the other nodes in area 7, and each of the nodes in areas 8 and 9.

Figure A–2 Example of Multiple-Area Network Design



A.3 Sample Multiple-Area Network Configuration

Figure A–3 Area 7 of a Multiple-Area Network



```
!
!  Define executor-specific parameters for local node A.
!  Note that the TYPE parameter for the executor node
!  defaults to either NONROUTING IV or ROUTING IV, depending
!  on whether an end node or full function key has been
!  installed.  In this example, node A needs to be a level 2
!  router, so the TYPE parameter is set accordingly.
!
 DEFINE EXECUTOR ADDRESS 7.1 -
                 BUFFER SIZE 576 -
                 STATE ON -
                 TYPE AREA -
                 DEFAULT PROXY BOTH
!
!  Define common node parameters for the local node.  Be
!  sure to add the NETNONPRIV user to your system
!  authorization file by using the Authorize Utility.
!
 DEFINE EXECUTOR NAME A -
                 NONPRIVILEGED -
                      USER NETNONPRIV -
                      PASSWORD NONPRIV -
!
!  Define the remaining nodes. Note that no default outbound
!  access control information is specified. This assumes that
!  the default access control information will be supplied by
!  each remote node when it receives an inbound connection, or
!  as a result of a proxy login on the target node.
!
 DEFINE NODE B ADDRESS 7.2
 DEFINE NODE C ADDRESS 7.3
 DEFINE NODE D ADDRESS 7.4
 DEFINE NODE E ADDRESS 7.5
 DEFINE NODE F ADDRESS 7.6
 DEFINE NODE G ADDRESS 7.7
 DEFINE NODE H ADDRESS 7.8
 DEFINE NODE I ADDRESS 7.9
 DEFINE NODE J ADDRESS 7.10
 DEFINE NODE K ADDRESS 7.11
 DEFINE NODE L ADDRESS 7.12
!
!  If node L is a Phase III node, it would be necessary
!  to specify routing initialization passwords to
!  initialize this node.  Using the number of the area in
!  which the Phase III node resides as part of the password
!  will avoid accidental connection to another area.
!  Define a receive password for node L as follows:
!     DEFINE NODE L RECEIVE PASSWORD AREA7
!  In this case, on node L, the transmit password would be
!  set to match:
!     DEFINE NODE A TRANSMIT PASSWORD AREA7   ! (on node L)
!
!  Note that although nodes M and N reside in different
!  areas, no special action is needed in defining them.
!  Continue defining nodes in other areas in this fashion.
!
 DEFINE NODE M ADDRESS 8.3
 DEFINE NODE N ADDRESS 9.42
```

```
!
!  Set up the line and circuit for the Ethernet connected
!  to node A.
!
 DEFINE LINE UNA-0 STATE ON
 DEFINE CIRCUIT UNA-0 STATE ON
!
!  Set up the line and circuit for each DMR connected to
!  node A. Note that a DMR line is treated like a DMC line.
!
 DEFINE LINE DMC-1 STATE ON
 DEFINE CIRCUIT DMC-1 STATE ON
 DEFINE LINE DMC-2 STATE ON
 DEFINE CIRCUIT DMC-2 STATE ON
!
!  Set up the line and circuit for the DMC connected to
!  node A.  Because the DMC leads to another area, you
!  may want to leave this circuit and line in the OFF state
!  while you are initially configuring your area, turning
!  them on only after the connections within your area
!  have been tested.
!
 DEFINE LINE DMC-0 STATE ON
 DEFINE CIRCUIT DMC-0 STATE ON
!
!  The object database does not need to be defined, since it
!  defaults to the standard list of objects known to VAX/VMS.
!
!  Define the transmitter-related logging parameters.
!
 DEFINE LOGGING MONITOR KNOWN EVENTS
!
!  Define receiver-related logging parameters.
!
 DEFINE LOGGING MONITOR STATE ON
```

A.4 Converting an Existing Network to a Multiple-Area Network

Converting an existing single-area network to a multiple-area network requires careful planning. Because the network addresses of existing nodes change, there may be a period during which some nodes are unreachable while the conversion is under way. The following steps provide an approach that can keep this disruption to a minimum:

1. Plan ahead. Completely define what the entire network topology will be with multiple areas. Make sure the topology follows the guidelines listed in Section A.1. Decide which nodes should be level 2 routers, level 1 routers, and end nodes.
2. If the new design requires some nodes to be moved, make the required changes before you begin converting node addresses. (For example, the redesign may involve reconnecting a Phase III node so that it is not in a path between two Phase IV nodes.)
3. Create new node databases. Without modifying the existing permanent node databases, create a new copy of the node database on each node in the network. For DECnet–VAX nodes, you can do this by following these steps:
   a. Use the logical name NETNODE_REMOTE to point to the working copy of the remote node file you are creating and use the logical name

NETNODE_LOCAL to point to the working copy of the local node file you are creating. These logical names will be translated when NCP is reading the permanent database, and the default versions of NETNODE_REMOTE.DAT and NETNODE_LOCAL.DAT in SYS$SYSTEM will remain untouched. Set up this environment as follows:

```
$ COPY SYS$COMMON:[SYSEXE]NETNODE_REMOTE.DAT -
_$ SYS$MANAGER:NEWNETNODE_REMOTE.DAT
$ COPY SYS$SPECIFIC:[SYSEXE]NETNODE_LOCAL.DAT -
_$ SYS$MANAGER:NEWNETNODE_LOCAL.DAT
$ ASSIGN/USER_MODE SYS$MANAGER:NEWNETNODE_REMOTE.DAT NETNODE_REMOTE
$ ASSIGN/USER_MODE SYS$MANAGER:NEWNETNODE_LOCAL.DAT NETNODE_LOCAL
$ RUN SYS$SYSTEM:NCP
NCP>
```

When the NCP prompt appears, enter the changes, which will be made to the new database.

b. In your remote node file, change the addresses of existing nodes in the network to reflect the new topology with areas. If you are adding new nodes to the network (for example, if two existing separate networks are to be merged), include their new addresses in your file now. Make sure these changes are only made to the working copies of the files, and not to SYS$SYSTEM:NETNODE_REMOTE.DAT or SYS$SYSTEM:NETNODE_LOCAL.DAT. For now, the database changes need to be made on only one node in the network. Also, to modify the executor, enter the following commands:

```
NCP>PURGE EXECUTOR ADDRESS
NCP>DEFINE NODE local__node__name ADDRESS ...
NCP>DEFINE EXECUTOR ADDRESS ...
```

c. When SYS$MANAGER:NEWNETNODE_REMOTE.DAT and SYS$MANAGER:NEWNETNODE_LOCAL.DAT correctly reflect the new topology, copy SYS$MANAGER:NEWNETNODE_REMOTE.DAT to the SYS$MANAGER directory on each node in the network. (Note that you should do this only for nodes that are running the same version of VMS. If you have different versions in the network, you should follow this conversion procedure independently for each version.)

On each VMS node on the network, define the executor parameters in NEWNETNODE_LOCAL.DAT, as follows:

```
$ COPY SYS$SPECIFIC:[SYSEXE]NETNODE_LOCAL.DAT -
_$ SYS$MANAGER:NEWNETNODE_LOCAL.DAT
$ ASSIGN/USER_MODE SYS$MANAGER:NEWNETNODE_REMOTE.DAT NETNODE_REMOTE
$ ASSIGN/USER_MODE SYS$MANAGER:NEWNETNODE_LOCAL.DAT NETNODE_LOCAL
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE EXECUTOR ADDRESS ...
```

Using the NCP command DEFINE EXECUTOR, set up each local node with the correct area and node address, correct executor type (nonrouting IV, routing IV, or area), and other executor parameters.

Similarly, convert the node database on each non-VMS node in your network using the tools available for each implementation.

4 Shut down the network and bring it up again with the new database.

This is the only part of the conversion process that benefits from real-time cooperation among the nodes in your network. If this level of coordination is not feasible, then avoid using area number 1 for any area in the revised network. Since the area number defaults to number 1 if none is specified, it is possible that node address duplications may occur during the transition period.

At approximately the same time, have each node in the network shut down DECnet. You do not need to shut down the operating system. Enter the following command:

```
NCP>SET EXECUTOR STATE SHUT
```

When all nodes have shut down DECnet (or after an agreed-upon interval during which all nodes should have shut down DECnet), rename the new copy of the database and restart the network at each node:

```
$ RENAME SYS$MANAGER:NEWNETNODE_REMOTE.DAT -
_$ SYS$COMMON:[SYSEXE]NETNODE_REMOTE.DAT
$ RENAME SYS$MANAGER:NEWNETNODE_LOCAL.DAT -
_$ SYS$SPECIFIC:[SYSEXE]NETNODE_LOCAL.DAT
$ @SYS$MANAGER:STARTNET
```

Note that if your node is on an Ethernet to which applications other than DECnet (such as LAT) are connected, these applications should also be shut down along with DECnet, and then restarted after DECnet is restarted. This step is necessary because DECnet will be changing the Ethernet physical address of your node to reflect the new executor node address (see Section 3.3.4.1).

5 Use NCP to monitor the reconfigured network.

Depending on the size of the network and the care with which the conversion was done, there may be a period of debugging the network to ensure that all desired connections have been made. You can simplify debugging the conversion if you can run each area separately for a while before connecting them. You can do this by turning the circuits between level 2 routers in different areas to the OFF state in the new copy of the database. When you are confident that an area is operating to your satisfaction, you can turn on the circuits joining this area to its neighboring areas. Of course, while the interarea circuits are off, nodes in those areas are not accessible to nodes in other areas. This circumstance may be viewed as a tradeoff to reduce the number of variables during the conversion.

A.5 Problems in Configuring a Multiple-Area Network

The use of area routing techniques for configuring a network can lead to certain problems that may not be readily identifiable. The following sections describe some problems related to violation of the area routing configuration guidelines presented in Section A.1, and explain how to solve these problems.

A.5.1 Partitioned Area Problem

Improper configuration of the network topology for a multiple-area network can result in a failure condition that can cause traffic to be incorrectly routed, lost, or both. The problem is called area partitioning; it occurs when an area is broken into separate parts as the result of the failure of one or more lines or nodes. As a consequence of partitioning, a node may be isolated within an area.

Figure A–4 illustrates an improper network design, in which an area is vulnerable to partitioning if a single line should fail. All circuit costs in Figure A–4 are equal to 1. Node C in area 3 attempts to communicate with node D in area 4. If either link *w* or *x* fail, no problem arises because the remaining path into area 4 provides a route to node D. If link *y* or *z* fail, the level 2 router in area 3 will find the path to the level 2 router in area 4 on the basis of the least-cost algorithm; the path would be from node C to node B to node A. Because link *y* or *z* is down, however, it is not possible to get to the destination node D.

When the initial connection is attempted, the network turns on the "return to sender" bit and sends a message to the sender indicating the node is unreachable. If the two nodes have already established a link before the connection breaks, the sender will time out, because the network will not route back traffic when it arrives at the destination area. Thus, a node in an area is isolated because of a line failure.

Figure A–4 illustrates another problem. If link *z* is down and node D wants to create a link to node B, the path that node D chooses is to route through area 2 and then to node B. On the return trip, however, node B will attempt to send the reply to node A, but link *z* is down, and therefore the reply will not be delivered to node D.

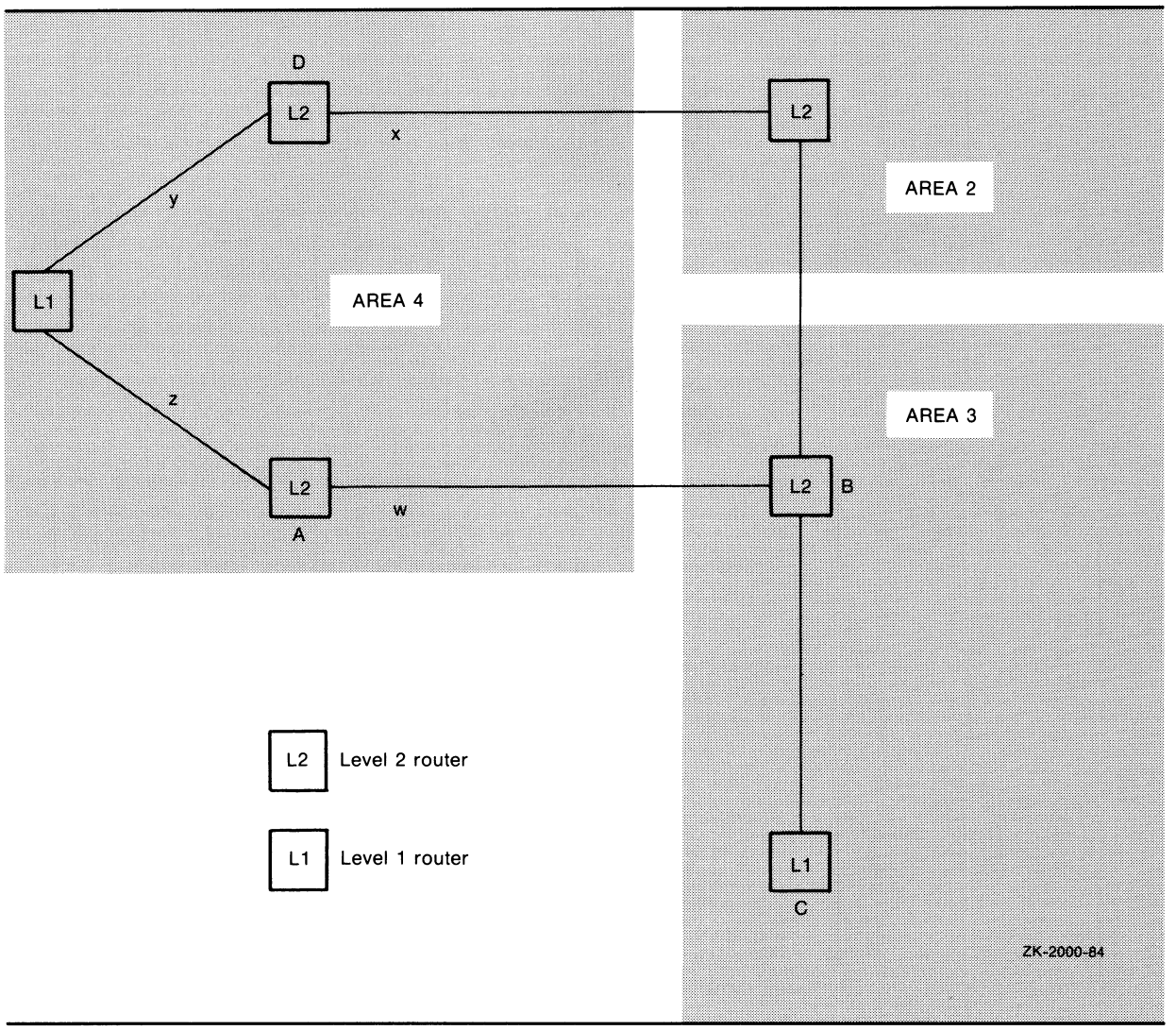
The solution to the problem of area partitioning is to treat each area as a separate network when configuring a multiple-area network. Designing an area as a straight-line configuration, as in area 4 in Figure A–4, should be avoided. Also, all the level 2 routers in a given area should be linked in a level 2 routing path; a level 1 router should not be included on the same path. In the configuration in Figure A–4, installing a link between nodes A and D would provide for an alternate path between nodes in the same area.

A.5.2 Problems in Mixed Phase III/Phase IV Networks

In a Phase IV multiple-area network, Phase III nodes can be included provided certain rules are followed: Phase III nodes in a multiple-area network must not be in the routing path between Phase IV nodes and must not be linked to nodes outside their own area. These limitations are based on the following restrictions under which Phase III nodes operate in a Phase IV network:

- DECnet–VAX Phase III nodes cannot have a node address greater than 255, and cannot directly address a node with a node address greater than 255. They also cannot route through traffic for nodes with addresses greater than 255. (Note that this limit may vary for other DECnet Phase III nodes, up to a maximum possible address of 1023.)

Figure A–4 Partitioned Area Problem



- Phase III nodes cannot recognize area numbers in node addresses. A Phase III node cannot assume an area address, directly address a node outside its own area, or route through traffic for nodes in other areas.
- Phase III nodes cannot be connected directly to an Ethernet.
- Phase III nodes must use routing initialization passwords when they are initialized in a Phase IV network (see Section A.5.2.2).

The node address is represented in different ways in Phases III and IV. In Phase III a node address is represented by a single decimal number, such as 99. For DECnet–VAX, the maximum node address of a Phase III node is 255. In Phase IV a node address is represented by a number in the following format:

area-number.node-number

where:

area-number	Is a maximum of 63.
node-number	Is a maximum of 1023.

If Phase IV node number 99 is in area 33, its node address is 33.99. (If a Phase IV network is not configured into areas, node number 99, by default, is in area number 1 and is represented in the database by the node address 1.99.)

Whenever a Phase III node is brought up in a Phase IV multiple-area network, the physical link is initialized with the Phase III protocol and all references to the area number are dropped. Routing in the network is affected in different ways, depending on the direction in which traffic is flowing:

- If traffic is going from a Phase IV node to the Phase III node, the area number is dropped from the node address. For example, when node address 19.201 is passed to a Phase III node, the node address becomes 201.
- If traffic is going from a Phase III node to a Phase IV node, the area number of the Phase IV node is added to the node address. For example, if node address 143 is sent to Phase IV node 75.5, the node address 143 becomes 75.143.
- If a packet is routed through a Phase III node, the area number is dropped from both the source and destination node addresses in the routing header of the packet.

A.5.2.1 Problem of a Phase III Node in a Phase IV Path

An example of the problem caused by placing a Phase III node in the routing path between two Phase IV nodes in the same area is illustrated in Figure A–5.

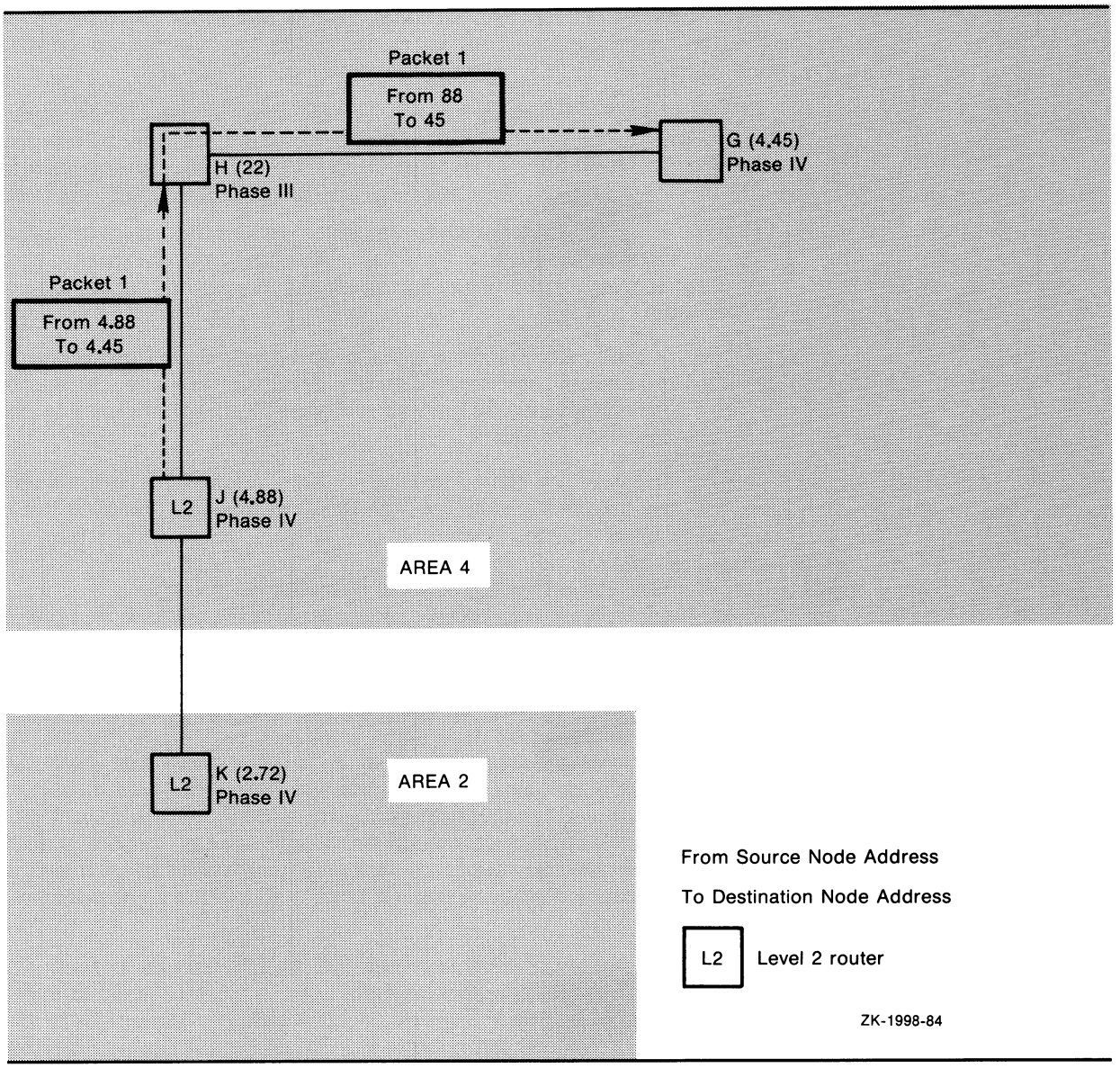
No problem occurs if the entire logical link path is within a single area and if none of the nodes have node numbers greater than 255. For example, for node 4.88 to send a packet to node 4.45, the packet first goes to node 22 (the Phase III node has no area number even though it is in area 4). Node 22 discards the area number from the destination node address 4.45, making it address 45, and from the source node address 4.88, making it address 88. The packet is then forwarded to node 4.45, which adds its own area number to the destination address, making it 4.45, and to the source address 88, making it 4.88.

During the return trip from node 4.45 to 4.88, the packet (with source address 4.45 and destination address 4.88) goes through node 22 and loses the area numbers from the source and destination node addresses in its routing header. When the packet arrives at its destination, node 4.88 adds its own area number to the node addresses in the routing header, making the source address 4.45 and the destination address 4.88.

A problem occurs, however, during communication between nodes in different areas, when the routing path in one area includes a Phase III node between two Phase IV nodes. In Figure A–5, if node 2.72 wants to send a packet to node 4.45, the packet goes from node 2.72 to node 4.88 and then to node 22, which drops the area number from both the source and destination addresses in the routing header, making the source address 72 and the destination address 45. Node 22 then sends the packet to the destination, node 4.45, which adds its area number to the source address, making it 4.72, and to the destination address, making it 4.45. The problem arises during the

return trip. Node 4.45 attempts to respond by sending a packet addressed to destination node 4.72 instead of 2.72. If a node with address 4.72 does exist, the return packet is incorrectly delivered to that node. Node 2.72 does not receive a reply and eventually times out.

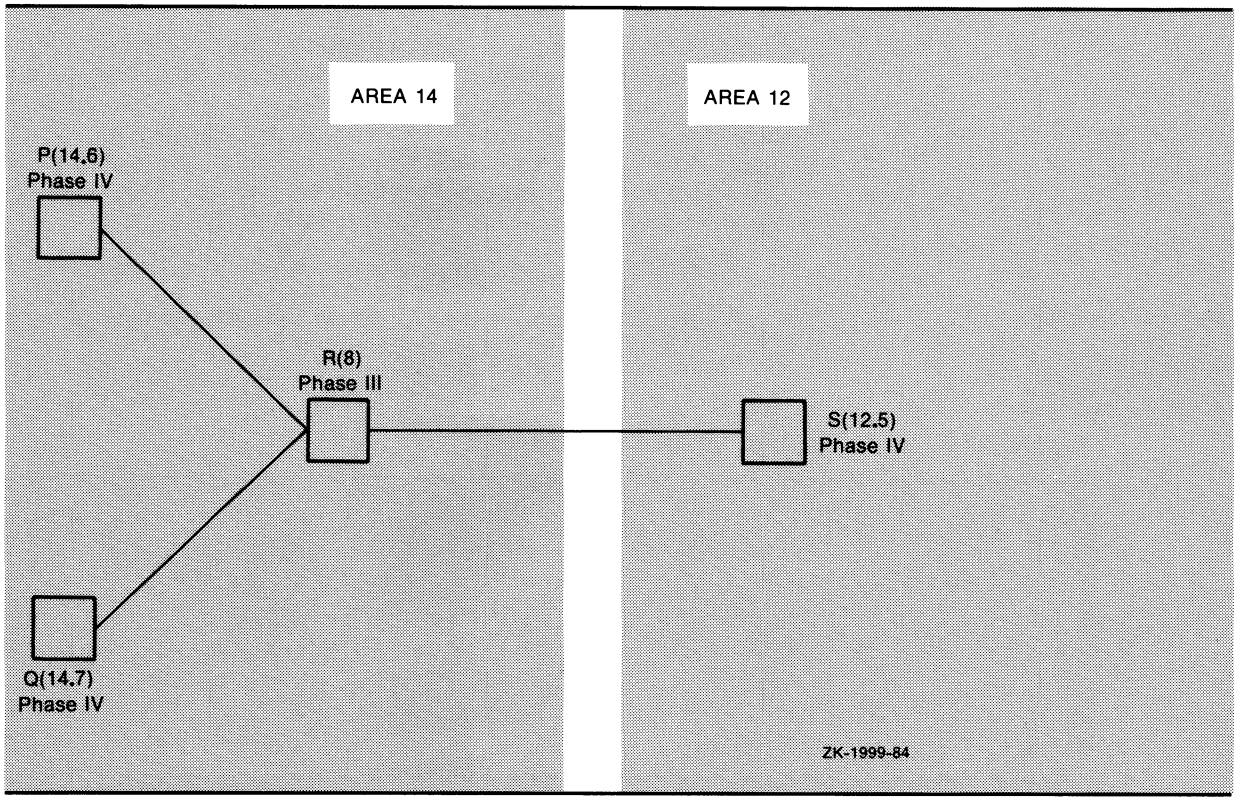
Figure A–5 Problem of Phase III Node In Phase IV Path



A.5.2.2 Area Leakage Problem

When a Phase III node is included in a Phase IV network that has been divided into multiple areas, the Phase III node should not be connected to a node outside its own area (as in Figure A–6). A Phase III node drops the area number from a node address. Permitting a Phase III node to have a link to another area causes a problem known as "area leakage." When the Phase III node builds its routing database, it includes the node addresses of adjacent nodes minus their area numbers. This incorrect information is then transmitted (or "leaked") across the area boundaries. This problem occurs whether the Phase IV nodes are level 1 or level 2 routers.

Figure A–6 Area Leakage Problem



In Figure A–6, node R is a Phase III node with links to nodes in areas 12 and 14. Node R will build a routing database that contains the addresses of nodes P, Q, and S, but the area numbers will be missing from the node addresses. Node R will send routing updates to all adjacent nodes, without recognizing area boundaries. Thus, node R will send routing information about nodes P and Q (minus the correct area designation) to node S. Node S will assume that nodes P and Q are nodes in its own area that have the addresses 12.6 and 12.7, respectively, instead of the correct addresses 14.6 and 14.7. Similarly, node R will send the address of node S (minus its area number) to nodes P and Q; nodes P and Q will assume that node S is in their own area and has the address 14.5 rather than the correct address 12.5.

Routing initialization passwords are required when a Phase III node is initialized in a Phase IV network (see Section 2.10.1 for a description of the passwords). If no password is specified during routing initialization, a specific event class message will be generated, indicating that a password is required or is mismatched. If the event logger is turned on, the network manager can read these messages to learn which Phase III nodes have not been initialized. The network manager can use this information to prevent Phase III nodes from linking to nodes outside their own areas, or to identify which Phase III nodes need to have the transmit password set. To prevent accidental connection to a node in a wrong area, the number of the area in which the node resides should be used in the password.

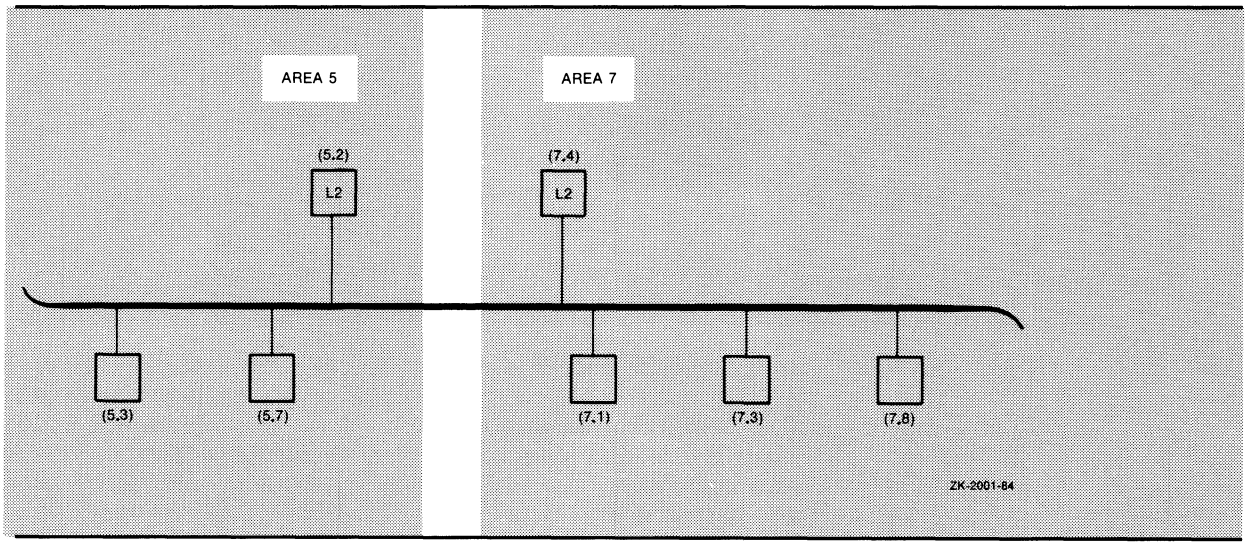
Note that this technique will not locate Phase III nodes improperly linked to nodes in other areas if the Phase III nodes were configured using routing initialization passwords before conversion to area routing, unless the passwords were changed as recommended during the conversion.

A.6 Area Routing on an Ethernet

Phase IV DECnet supports configuration of multiple areas on an Ethernet. This configuration results in higher message overhead because a packet must be routed through two level 2 routers rather than be delivered directly to the destination node. Figure A–7 shows two areas sharing the same Ethernet cable. When a node in area 5 wants to communicate with a node in area 7, the packet is routed through the two level 2 routers. For example, if node 5.3 sends a packet to node 7.8, the packet follows this path:

- Node 5.3 to node 5.2 (source to its nearest level 2 router)
- Node 5.2 to node 7.4 (level 2 router to level 2 router)
- Node 7.4 to node 7.8 (level 2 router to destination)

Figure A–7 Area Routing on an Ethernet



Glossary

access control: Validating connect, login, or file-access requests to determine whether they can be accepted. User name and password provide the most common means of access control.

account name: A string that identifies a particular account used to accumulate data on a job's resource use. This name is the user's accounting charge number, not the user's UIC.

active component: A component whose operational state is other than OFF. You can use the word ACTIVE with the SHOW or LIST command to display information about active lines, circuits, nodes, and logging.

adjacent node: A node removed from the local node by a single physical line.

alias node identifier: An optional node name or address, common to some or all nodes in a VAXcluster, that permits the VAXcluster to be treated as a single node.

area: A group of nodes in a network that can run independently as a subnetwork.

area router: A level 2 router.

area routing: A technique for grouping the nodes in a network into areas for routing purposes. Routing in a multiple-area network is hierarchical, with one level of routing within an area (level 1 routing) and a second, higher level of routing between areas (level 2 routing).

asynchronous transmission: A mode of data transmission in which the time intervals between transmitted characters may be of unequal length. Asynchronous transmission most commonly occurs over terminal lines.

bandwidth: The range of frequencies assigned to a channel or system (that is, the difference expressed in Hertz between the highest and lowest frequencies of a band).

bilateral closed user group (BCUG): An optional packet switching data network (PSDN) facility that restricts a pair of DTEs from communicating with each other.

broadcast addressing: A special type of multicast addressing, in which all nodes are to receive a message.

broadcast circuit: A circuit on which multiple nodes are connected and on which a message can be transmitted to multiple receivers.

carrier sense: A signal provided by the Physical layer to indicate that one or more stations (nodes) are currently transmitting on the Ethernet channel.

Carrier Sense, Multiple Access with Collision Detect (CSMA/CD): A link management procedure used by the Ethernet. Allows multiple stations to access the broadcast channel at will, avoids contention by means of carrier sense and deference, and resolves contention by means of collision detection and retransmission.

CCITT: Comite Consultatif International Telegraphique et Telephonique. An international consultative committee that sets international communications usage standards.

channel: A means of transmission. For VAX PSI, a logical path between a DTE and a DCE over which data is transmitted. Each channel is identified by a unique reference number called a logical channel number (LCN).

characteristics: A display type for the SHOW and LIST commands. It refers to static information about a component that is kept in either the volatile or permanent database. Such information may include parameters defined for that component by either the SET or DEFINE command.

circuit: Virtual communication path between nodes or DTEs. Circuits operate over physical lines and are the medium on which all I/O occurs. X.25 circuits are virtual circuits.

closed user group (CUG): An optional PSDN facility that restricts two or more DTEs in the same group from communicating with each other.

collision: Multiple transmissions overlapping in the physical channel, resulting in garbled data and necessitating retransmission.

collision detect: A signal provided by the Physical layer to the Data Link layer to indicate that one or more stations (nodes) are contending with the local station's transmission.

command node: The node from which an NCP command is entered.

component: An element in the network that can be controlled and monitored. Components include lines, circuits, nodes, modules, logging, and objects. Components form part of the NCP command syntax.

configuration database: The combination of both the permanent and the volatile databases. It consists of information about the local node, and all nodes, modules, circuits, lines, and objects in the network.

congestion loss: A condition in which data packets are lost when Routing is unable to buffer them.

connector node: A node which serves as an X.25 gateway to permit VMS host nodes to access a packet switching data network.

control station: The node at the controlling end of a multipoint circuit. The control station controls the tributaries for that circuit.

cost: An integer value assigned to a circuit between two adjacent nodes. According to the routing algorithm, packets are routed on paths with the lowest cost.

counters: Performance and error statistics kept for a component, such as lines or nodes.

data circuit-terminating equipment (DCE): A CCITT X.25 term referring to the network equipment that establishes, maintains and terminates a connection and handles the signal conversion and coding between the data terminal equipment and the network. The switching exchange of the network to which DTEs are connected. (In non-X.25 usage, the term is synonymous with *modem*.)

data link mapping (DLM): Capability of using an X.25 virtual circuit as a DECnet data link.

data terminal equipment (DTE): An X.25 term referring to the user's equipment (computer or terminal) connected to a DCE on a packet switching data network for the purpose of sending and receiving data.

datagram: A unit of data sent over the network that is handled independently of all other units of data as far as the network is concerned. When a route header is added, a datagram becomes a packet.

designated router: A routing node on the Ethernet selected to perform routing services on behalf of end nodes.

disconnect abort: A method by which nontransparent tasks can deaccess a logical link by means of a disconnect abort operation without deassigning the channel. This form of disconnection indicates to the receiver that not all messages sent have necessarily been received.

downline system load: A DECnet–VAX function that allows an unattended target node to receive an operating system file image from another node.

downline task load: A function that allows a remote target node to receive an RSX–11S task from another node.

end node: A node that can receive packets addressed to it and send packets to other nodes, but cannot route packets through from other nodes. Also called a nonrouting node.

equal cost path splitting: The process by which a packet load is split for routing over multiple equal cost paths to a destination node.

event: A network or system-specific occurrence for which the logging component maintains a record.

event class: A particular classification of events. Generally, this classification follows the DNA architectural layers; some layers may contain more than one class. Class also includes the identification of system-specific events.

event type: A particular form of event that is unique within an event class.

executor node: The node at which an NCP command actually executes.

frame: A unit delimited by flags that includes a header, and is used by the link level to exchange packets as well as control and error information between the DTE and the DCE on a packet switching data network.

handshaking sequence: The exchange of logical link connection information between two tasks. This exchange takes place to enable the successful completion of a logical link connection.

hardware address: For an Ethernet device, the unique Ethernet physical address associated with a particular Ethernet communications controller (usually in read-only memory) by the manufacturer.

hop: The logical distance between two nodes. One hop is the distance from one node to an adjacent node.

host node: For DECnet, a node that provides services for another node (for example, the host node supplies program image files for a downline load).

For VAX PSI, a node that accesses a packet switching data network by means of an X.25 multihost connector node.

inbound connection: Refers to the fact that a task receives logical link connection requests.

interrupt: For VAX PSI, a packet, sent through a PSDN, that bypasses normal flow control procedures used by data packets.

interrupt message: During nontransparent task-to-task communication, a user-generated message sent outside the normal exchange of data messages. This usage of the term *interrupt* is contrary to the normal usage, which means to designate a software or hardware interrupt mechanism.

known component: The classification for one or more of the same components. This classification includes all active and inactive occurrences of the component type. For example, known nodes include all active and inactive nodes in the network.

level 1 router: A node that can send and receive packets, and route packets from one node to another, only within a single area.

level 2 router: A node that can send and receive packets, and route packets from one node to another, within its own area and between areas. Also known as an area router.

line: The network management component that provides a distinct physical data path.

Link Access Protocol (LAP): A set of procedures used for link control. X.25 defines two sets of procedures:

- LAP—The DTE/DCE interface is defined as operating in two-way simultaneous Asynchronous Response Mode (ARM) with the DTE and DCE containing a Primary and Secondary function.
- LAPB—The DTE/DCE interface is defined as operating in two-way Asynchronous Balanced Mode (ABM).

In addition, LAPB with extended sequence numbering (that is, frame numbering modulo 128) is known as LAPBE.

load assist agent: An image that provides additional data required to perform a downline system load to a node in a Local Area VAXcluster.

local node: The node at which you are physically located.

logical channel: A logical link between a DTE and its DCE. The physical communications line between a DTE and DCE is divided into a set of logical channels.

logical channel number (LCN): A unique reference number that identifies a logical channel. A DTE recognizes a virtual circuit by its associated LCN.

logical link: A carrier of a single stream of full-duplex traffic between two user-level processes.

logging: The network management component that routes event data to a logging sink such as a console or file.

logging console: A logging sink that is to receive a human-readable record of events. Typically, a logging console is a terminal or a user-specified file.

logging file: A logging sink that is to receive a machine-readable record of events for later retrieval. The logging file is user defined.

logging monitor: A logging sink that is to receive a machine-readable record of events for possible real-time decision making. Typically, the logging monitor is a user-defined program.

loop node: A local node that is associated with a particular line and is treated as if it were a remote node. All traffic to the loop node is sent over the associated line.

maximum visits: The maximum number of nodes through which a packet can be routed before reaching its destination.

module: A network management component.

multiaccess channel: A medium (for example, Ethernet) on which many transmitters contend for access.

multicast addressing: An addressing mode in which a given message packet is targeted to a group of logically related nodes.

multicast group address: An address assigned to a number of nodes on an Ethernet and used to send a message to all nodes in the group in a single transmission.

multipoint circuit: A circuit connecting two systems, with one of the systems (the control station) controlling the circuit, and the other system serving as a tributary.

network connect block (NCB): For DECnet, a user-generated data structure used in a nontransparent task to identify a remote task and optionally send user data in calls to request, accept, or reject a logical link connection.

For VAX PSI, a block that contains the information necessary to set up an X.25 virtual circuit or to accept or reject a request to set up an X.25 virtual circuit.

network status notifications: Notifications that provide information about the state of both logical and physical links over which two tasks communicate. A nontransparent task can use this information to take appropriate action under conditions such as third-party disconnections and a partner's exiting before I/O completion.

network task: A nontransparent task that is able to process multiple inbound connection requests; that is, it has declared a network name or object number.

node: A network management component that supports DECnet software.

node address: The required, unique, numeric identification of a specific node in the network.

node name: An optional alphanumeric identification associated with a node address in a strict one-to-one mapping. A node name must contain at least one alphabetic character.

nonprivileged: In DECnet–VAX terminology, means no privileges other than NETMBX and TMPMBX. NETMBX is the minimal requirement for any network activity.

nonrouting node: An end node.

object: A DECnet–VAX process that receives a logical link request. It performs a specific network function (a nonzero object such as FAL or NML), or is a user-defined image for a special-purpose application (a zero-numbered object).

A VAX PSI management component that contains records to specify account information for incoming calls and to specify a command procedure that is initiated when the incoming call arrives.

outbound connection: Refers to the fact that a task sends logical link connection requests.

out-of-order packet caching: The mechanism by which the Network Services Protocol (NSP) maintains a buffer of data packets received out of order so that they can be reassembled in the correct order before being forwarded to the destination node.

packet: A unit of data to be routed from a source node to a destination node. For VAX PSI, the unit of data switched through a PSDN; normally a user data field accompanied by a header carrying destination and other information.

packet assembly/disassembly (PAD) device: A device at a PSDN node that allows access from an asynchronous terminal. The terminal connects to the PAD and the PAD puts the terminal's input data into packets (assembles) and takes the terminal's output data out of packets (disassembles).

packet switching: A data transmission process, using addressed packets, whereby a channel is occupied only for the duration of transmission of the packet.

packet switching data network (PSDN): A set of equipment and interconnecting links that provides a packet switching communications service to subscribers.

Packetnet System Interface (PSI): The name for the software product that allows DIGITAL operating systems to participate in a packet switching environment.

parameter: An entry in the volatile or permanent database for a network management component.

path: The route a packet takes from source to destination.

path cost: The sum of the circuit costs along a path between two nodes.

path length: The number of hops along a path between two nodes; that is, the number of circuits along which a packet must travel to reach its destination.

permanent database: A file containing information about network management components.

permanent virtual circuit (PVC): A permanent logical association between two DTEs, which is analogous to a leased line. Packets are routed directly by the network from one DTE to the other.

physical address: The unique address value associated with a given system on an Ethernet circuit. An Ethernet physical address is defined to be distinct from all other physical addresses on an Ethernet.

point-to-point circuit: A circuit that connects two nodes, operating over a single line.

polling: The activity that the control station performs with a multipoint circuit's tributaries to grant the tributaries permission to transmit.

privileged: In DECnet–VAX terminology, means any user privileges in addition to NETMBX and TMPMBX.

protocol: An agreed set of rules governing the operation of a communications link.

proxy login: The procedure that permits a remote user to access a specific account at the local node, without supplying the user name and password.

reachable node: A node to which the local node has a usable communications path.

remote DTE: Any DTE in a network other than the one at which the user is located.

remote node: To any one node in the network, any other network node.

router: A node that can send and receive packets, and route packets from one node to another.

routing: The network function that determines the path along which data travels to its destination.

routing node: A router.

sink node: A node where logging sink types, such as a file or console, are actually located.

source task: The task that initiates a logical link connection request in a task-to-task communication environment.

state: The functions that are currently valid for a given component. States include line, circuit, local node, module, DTE, and logging.

status: A display type for the SHOW and LIST commands. Status refers to dynamic information about a component that is kept in either the volatile or permanent database.

substate: An intermediate circuit state that is displayed for a circuit state display by means of the SHOW or LIST command.

summary: The default display type for the SHOW and LIST commands. A summary includes the most useful information for a component, selected from the status and characteristics information.

switched virtual circuit (SVC): A temporary logical association between two DTEs connected to a PSDN, which is analogous to connection by a dialup line. An SVC is set up only when there is data to transmit and is cleared when the data transfer is complete.

synchronous disconnect: The disconnect that occurs when a nontransparent task issues a call to terminate I/O operations over a logical link without deassigning the channel. Thus, the task can use the channel for subsequent I/O operations with the same or a different remote task.

synchronous transmission: A mode of data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame.

target node: The node that receives a memory image during a downline load; a node that loops back a test message.

target task: The task that receives and processes a logical link connection request in a task-to-task communication environment.

task: In this manual, refers to an image running in the context of a process.

task specifier: Information provided to DECnet–VAX software so that it can complete a logical link connection to a remote task. This information includes the name of the remote node on which the target task runs and the name of the task itself.

terminal emulator: A program that acts as a transparent interface between two ports, making it appear as though a terminal on the local processor is directly connected to a remote processor.

tributary: A physical termination on a multipoint circuit that is not a control station.

tributary address: A numeric address that the control station uses to poll a tributary.

upline dump: A DECnet–VAX function that allows an adjacent unattended node to dump its memory to a file on a VMS operating system.

virtual circuit: An association between two nodes (or two DTEs connected to a PSDN) whereby the two nodes (or DTEs) are able to interact as if a specific circuit were dedicated to them throughout the transmission. When a virtual circuit is established, a logical connection is established, with the actual physical circuits being allocated according to route availability, overload conditions, and other factors.

virtual terminal: A pseudodevice that connects a process to a physical terminal device. The virtual terminal can be disconnected from the physical terminal and reconnected later.

volatile database: A memory image that contains information about network management components.

window: A range of packets authorized for transmission across an X.25 DTE/DCE interface. The lowest sequence number in the window is referred to as the lower window edge (0 when the virtual circuit is just established). The packet send sequence number of the first data packet not authorized to cross the interface is the value of the upper window edge (that is, the lower window edge plus the window size).

X.3: A CCITT recommendation that specifies the packet assembly/disassembly (PAD) facility in a public data network.

X.25: A CCITT recommendation that specifies the interface between data terminal equipment and data circuit-terminating equipment for equipment operating in the packet mode on public data networks.

X.28: A CCITT recommendation that specifies the DTE/DCE interface for a start-stop mode DTE accessing the packet assembly/disassembly (PAD) facility in a public data network situated in the same country.

X.29: A CCITT recommendation that specifies procedures for the exchange of control information and user data between a packet-mode DTE and a packet assembly /disassembly (PAD) facility.

X.29 terminal: A terminal connected to a packet assembly/disassembly (PAD) facility.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Reader's Comments

VMS Networking Manual
AA–LA48A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- Do Not Tear - Fold Here and Tape ----------

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------

Cut Along Dotted Line

Reader's Comments

VMS Networking Manual
AA–LA48A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

— Do Not Tear - Fold Here and Tape —

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

— Do Not Tear - Fold Here —

Cut Along Dotted Line