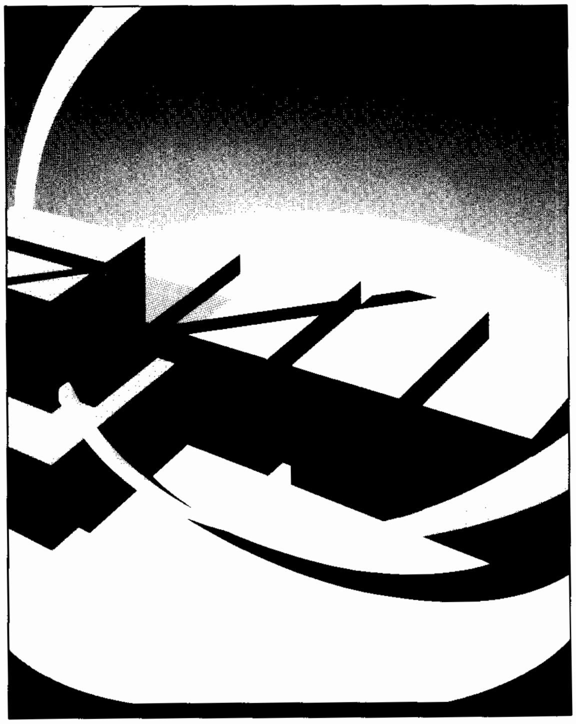
TANDEM SYSTEMS REVIEW

VOLUME 4, NUMBER 1 FEBRUARY 1988

BRANDIFINC



Tandem's NonStop SQL Benchmark

MULTILAN

Sizing the Spooler Collector Data File

C00 TMDS Performance

The C00 Release

Optical Storage

Fault Tolerance

TANDEM SYSTEMS REVIEW

Volume 4, Number 1, February 1988

Editor
Ellen Marielle-Tréhoüart

Associate Editors
Wendy Osborn
Carolyn Turnbull White

Assistant Editor
Sarah Rood

Technical Advisors
Mark Anderton
Bart Grantham

Cover Art
Janet Stevenson

Production and Layout
Corporate Graphics

Typesetting
Tandem Typography

The *Tandem Systems Review* is published by Tandem Computers Incorporated.

Purpose: The *Tandem Systems Review* publishes technical information about Tandem software releases and products. Its purpose is to help programmer-analysts who use our computer systems to plan for, install, use, and tune Tandem products.

Subscription additions and changes: Subscriptions are free. To add names or make corrections to the distribution database, requests within the U.S. should be sent to Tandem Computers Incorporated, *Tandem Systems Review*, 18922 Forge Drive, LOC 216-05, Cupertino, CA 95014. *Requests outside the U.S. should be sent to the local Tandem sales office.*

Comments: The editors welcome suggestions for content and format. Please send them to the *Tandem Systems Review*, 18922 Forge Drive, LOC 216-05, Cupertino, CA 95014.

Tandem Computers Incorporated makes no representation or warranty that the information contained in this publication is applicable to systems configured differently than those systems on which the information has been developed and tested. It also assumes no responsibility for errors or omissions that may occur in this publication.

Tandem's NonStop SQL Benchmark

Tandem's NonStop SQL™ is an implementation of the ANSI SQL relational database language. In contrast to other SQL implementations, NonStop SQL is designed for on-line transaction processing as well as for information-center use. It delivers high performance cost-effectively and supports distributed data with local autonomy.

This article describes a recent benchmark of NonStop SQL, which demonstrated over 200 transactions per second (tps).

The Purpose of the Benchmark

The NonStop SQL performance assurance group compared the throughput of ENSCRIBE™, Tandem's file access method, and the new SQL system. The benchmark was done jointly by Software Development in Cupertino, the High Performance Research Center in Frankfurt, and the Benchmark Center in Sunnyvale.

Parts of the benchmark were audited and certified by an independent auditor.

The benchmark demonstrated the following aspects of NonStop SQL:

- It is functional and has been stress-tested for high-volume on-line transaction processing (OLTP) applications.
- It allows distributed data and distributed transactions.
- NonStop SQL runs on small departmental systems as well as on large mainframe systems.
- It demonstrates linear increases in throughput when the disks and processors are added using the DYNABUS or the FOX™ fiber-optic ring.
- There is no apparent limit to the transaction throughput of NonStop SQL systems. In particular, there are no bottlenecks in systems running hundreds of transactions per second.
- NonStop SQL performs as well as the record-at-a-time ENSCRIBE system on OLTP applications.
- Both ENSCRIBE and NonStop SQL have impressive price/performance at both low and high transaction volumes.
- The price/performance of Tandem systems is competitive for both departmental systems (NonStop EXT10™) and data center systems (NonStop VLX™).

The Definition of Throughput and Price/Performance

The benchmark was based on the debit-credit banking application also known as ET1 (*Datamation*, April 1985). The article, "A Measure of Transaction Processing Power," defines a standard database, a standard terminal network, and a way to scale them to larger systems. It describes a standard transaction, called the debit-credit transaction, and specifies how to measure the throughput and price/performance of the resulting system.

Briefly, the database consists of four SQL tables:

- *Account*: a table of bank accounts. Each record is 100 bytes long and holds the account number and balance.
- *Teller*: a table of tellers. Each record is 100 bytes long and holds the teller number and cash position.
- *Branch*: a table of branches. Each record is 100 bytes long and holds the branch number and cash position of all tellers at the branch.
- *History*: an entry sequence table containing a list of all transactions. Each record is 50 bytes long and holds the account, teller, branch, delta, and timestamp.

The transaction, coded in SQL, is shown in Figure 1.

The system is presented with various transaction rates and the response time is measured over 10-minute windows, creating a response-time curve. (See Figure 2.)

A system that can run one such transaction per second, giving less than 1-second response time to 95% of the transactions, is called a 1-tps system. The database of a 1-tps system is defined as follows:

100,000	Accounts
100	Tellers
10	Branches
2,590,000	History

The History table is sized to accommodate 90 days of history records, assuming the average rate is one-third of the peak transaction rate.

Figure 1

```
READ 100 BYTES FROM TERMINAL;
BEGIN WORK;
PERFORM PRESENTATION SERVICES GIVING account_number,
                                      teller_number,
                                      branch_number,
                                      delta;
UPDATE ACCOUNT
       SET balance = balance + :delta
       WHERE account_number = :account_number
       and balance + :delta > =0
UPDATE TELLER
       SET balance = balance + :delta
       WHERE teller_number = :teller_number;
UPDATE BRANCH
       SET balance = balance + :delta
       WHERE branch_number = :branch_number;
INSERT INTO HISTORY VALUES
    (:timestamp,:account_number,:teller_number,:branch_number,:delta);
PERFORM PRESENTATION SERVICES;
COMMIT WORK;
WRITE 200 BYTES TO TERMINAL;
```

Figure 2

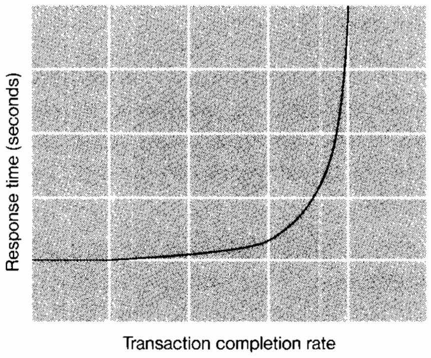


Figure 1.

Profile of the debit-credit transaction.

Figure 2.

A typical response-time curve showing how response time grows as the transaction load on the system increases.

The standard also specifies that a 1-tps system has 100 tellers at 10 branches. Each teller thinks for an average of 100 seconds and then submits a transaction. The tellers use block-mode terminals with ten fields of input and output. The input message is 100 bytes, and the output message is 200 bytes. Messages are transmitted via the X.25 communication protocol.

Systems that run more than 1 tps have the database sizes and network scaled linearly. For example, a 100-tps system has a database and teller network 100 times larger.

The standard specifies that accounts belong to branches and that tellers belong to branches. If the database is distributed, then 15% of the transactions arrive at branches other than the account's home branch. These 15% are uniformly distributed among the other branches.

The benchmark requires that the transactions be run with undo/redo transaction protection (abort, auto-restart, and rollforward recovery). In addition, it specifies that the transaction log must be duplexed.

Departures from the Standard Debit-Credit Transaction

The NonStop SQL benchmark departed from the *Datamation* standard definition in the following ways:

1. Terminals were driven in record mode (Intelligent Device Support) rather than block mode. In effect, this assumes that presentation services are done in the terminal.
2. It was assumed that each branch had a concentrator that multiplexed the teller terminals at the branch. This had the effect of reducing the number of sessions by a factor of ten for the same transaction rate when compared to the standard.
3. Tps was measured as the throughput when 90% of the transactions get 2-second (or less) response time. The standard measures tps at 1 second (or less) for 95% of the transactions.
4. The SQL software verified that debits did not cause negative account balances.
5. The system was measured over 10-minute periods, and all transactions for each period were used to compute the response-time curves and consequent tps ratings.
6. Response times were measured within the driver system. The standard specifies that response can be measured at the interface to the service system.
7. All devices were driven by NonStop processes so that no single failure would cause a denial of service.
8. All disks were mirrored (duplexed), as is standard with Tandem equipment.
9. Standard products were used. All applications were written in COBOL85.

Items 1, 2, and 3 tend to create an optimistic tps rating. Items 4 through 9 tend to reduce the system's tps rating.

The Benchmark System Design

The benchmark hardware comprised 32 VLX processors (32VLX). Each VLX processor had a 5-Mbyte-per-second channel and 8 Mbytes of main memory and was rated at about 3 Tandem MIPS (million instructions per second).

In addition, a NonStop EXT10 system was included in the benchmark hardware to demonstrate scaleability. At the time of the benchmark, the EXT10 was Tandem's smallest available NonStop™ system. It consists of two processors, each with 4 Mbytes of main memory, and four disks. The EXT10 processors together are approximately half as powerful as a VLX CPU. Thus, the complex was sized as a 32.5-processor VLX system. (See Figure 3.)

The VLX processors were divided into four nodes of eight VLX CPUs each. Each node connected its eight processors via dual 20-Mbyte-per-second DYNABUS interprocessor buses, and the nodes were interconnected via FOX, a dual, bidirectional fiber-optic ring. (See Figure 4.)

To model a distributed node, the EXT10 was connected to the VLX processors with a 9.6-Kbit communication line.

The GUARDIAN 90XF™ system made the entire 32VLX plus EXT10 configuration appear to be a single system with location transparency.

The database and transaction load were partitioned into 33 parts. Based on preliminary tests, each VLX processor would process less than 8 tps, and the EXT10 could process 4 tps. All sizings were based on these preliminary measurements.

Each transaction input message was 100 bytes long; the reply was 200 bytes long. With X.25 overheads, this translates to 340 bytes in all. At 8 tps, this is 22 Kbits per second. A 56-Kbit line can handle this load quite comfortably. Therefore, each VLX processor was configured with a 56-Kbit line, and the EXT10 was configured with a single 56-Kbit line to the driver system.

The database was sized as follows. Each eight-processor VLX (8VLX) node had:

- A mirrored disk to store the programs and the transaction log (audit trail).
- A mirrored disk dedicated to the History table. In the benchmark, only one volume was configured, but in pricing the system, the 90-day history file of each node was sized at 6.7 Gbytes (14 mirrored volumes).

According to the standard, 8 tps implies:

80 branches	8 Kbytes
800 tellers	80 Kbytes
800,000 accounts	80 Mbytes

Figure 3

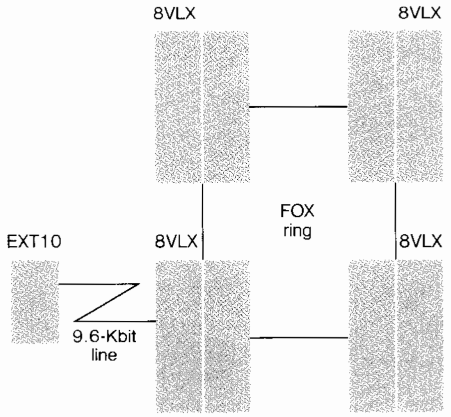
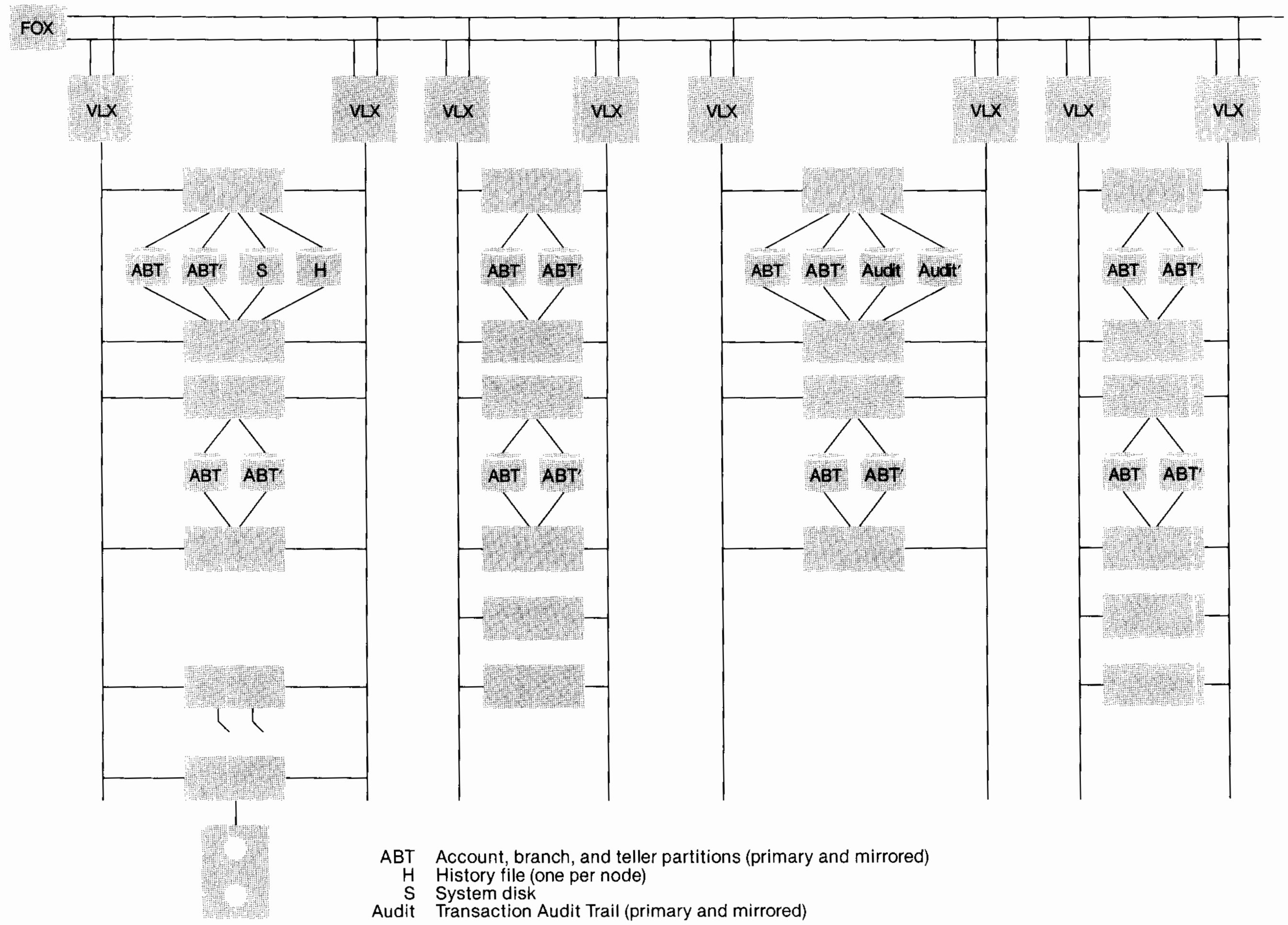


Figure 3.
The benchmark configuration. The two-processor EXT10 is connected to one VLX via a 9.6-Kbit line. The 32VLX processors are divided into four nodes of eight processors each. The nodes are connected by the high-speed FOX ring. The whole complex looks like a single system to the application programmer. A separate driver system submits transactions to this system via 56-Kbit X.25 lines.

The EXT10 system was sized at half of a VLX processor. The programs and audit trail were on one mirrored disk and the database resided on a second mirrored disk.

These records, along with their indexes and some slack, fit comfortably on Tandem's smallest disk. Therefore, each VLX processor was given a mirrored disk volume to hold its part of the Account-Branch-Teller (ABT) tables. A 1.6-Mbyte disk cache per mirrored disk partition was sufficient to keep all branches, tellers, and the account index pages resident in main memory.

Figure 4



ABT Account, branch, and teller partitions (primary and mirrored)
H History file (one per node)
S System disk
Audit Transaction Audit Trail (primary and mirrored)

Figure 4.

The configuration of each eight-processor VLX node in the FOX ring.

Figure 5.

Network and database sizes for the 208-tps benchmark system.

Figure 5

	Database	
Network	Records	Size
25,600 Tellers	25,600 Tellers	2.6 Mbytes
2560 Branches	2560 Branches	0.3 Mbytes
	25,600,000 Accounts	2.6 Gbytes
	54,000,000,000 History	27.0 Gbytes

Figure 5 shows the overall system configuration. The teller, branch, and account files were mirrored, but for lack of hardware, the history file was not mirrored.

The SQL tables were defined to be partitioned by the appropriate key values. The approximate syntax for creating a single Account table with 33 partitions is shown in Figure 6.

NonStop SQL hides this partitioning from the application programmer. The branch and teller files were partitioned in a similar way.

NonStop SQL imposes no practical limit on the number of partitions a table may have. Because ENSCRIBE is limited to a maximum of 16 partitions, the ENSCRIBE database and ENSCRIBE benchmark were limited to 16 VLX processors.

In the Tandem system, terminal control and presentation services are handled by a process called the Terminal Control Program (TCP). It is the logical equivalent of IMS/DC or CICS. Since each CPU was sized at 8 tps, each CPU supported 800 different teller records at 80 branches. It was assumed that each branch had a concentrator that multiplexed the teller terminals at the branch. This had the effect of reducing the number of sessions by a factor of 10 for the same transaction rate when compared to the standard. It was decided to configure 32 branches (320 tellers) per TCP, which resulted in approximately 2.5 TCPs per VLX CPU.

To avoid queuing on database servers at 8 tps with a 2-second average response time, 20 debit-credit servers were configured per CPU (20 ~ 2*8). The code and data for the TCPs was approximately 0.5 Mbytes per VLX CPU. The ENSCRIBE servers consumed 0.25 Mbytes per VLX. The NonStop SQL servers used approximately 10 Kbytes more memory—altogether 20 NonStop SQL servers used about 0.5 Mbytes per VLX.

A second complex of 12 NonStop TXP™ processors simulated the network of 2560 branches (25,600 tellers) by submitting transactions via X.25 using modem eliminators to connect the X.25 lines. Typically, transactions were submitted to each VLX line at an average rate between 4 tps and 8 tps, with exponentially distributed interarrival times. The EXT10 was driven at about 4 tps. Each transaction message named an account, branch, teller, and debit amount. As specified by the standard, in 85% of the cases the account and branch were local; in 15% of the cases the account was in a branch other than the branch and teller of this transaction. Some of these "nonlocal" transactions were in branches at the same node, but most went to other nodes of the network. For example, when the system was running at over 200 tps, the EXT10 might originate or receive 1 distributed transaction per second, while each VLX node might originate or receive 15 distributed transactions per second.

The driver system measured response time as the time between the send and the completion of the receive in the driver system. (See Figure 7.)

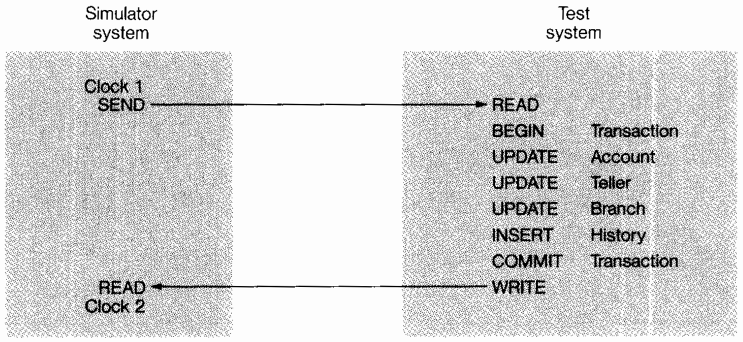
Figure 6

```
CREATE TABLE account (number    PIC 9(12),
                      balance   PIC S9(11)V(2),
                      ...
                      KEY number
                      )
                      PARTITION $vlx2 FIRST KEY      800000
                      PARTITION $vlx3 FIRST KEY     1600000
                      ...
                      PARTITION $vlx32 FIRST KEY 24800000
                      PARTITION $ext    FIRST KEY 25600000;
```

Figure 6.

The SQL statement used to create the account file partitioned among 33 disks.

Figure 7



Measured response time = Clock 2 – Clock 1

Note: The standard specifies the response time as the elapsed time between the last bit entering the test system until the time the first bit leaves the system.

Figure 7.

The driver system submits messages via X.25 and measures the transaction completion rate and response time.

Table 1.
Benchmark hardware configuration.

Driver system	Lines	Processors	Storage
24 MIPS	33*56 Kbit X.25	32VLX + EXT10 100 MIPS 264 Mbytes	86 disks 20 Gbytes + 27 Gbytes history (when pricing)

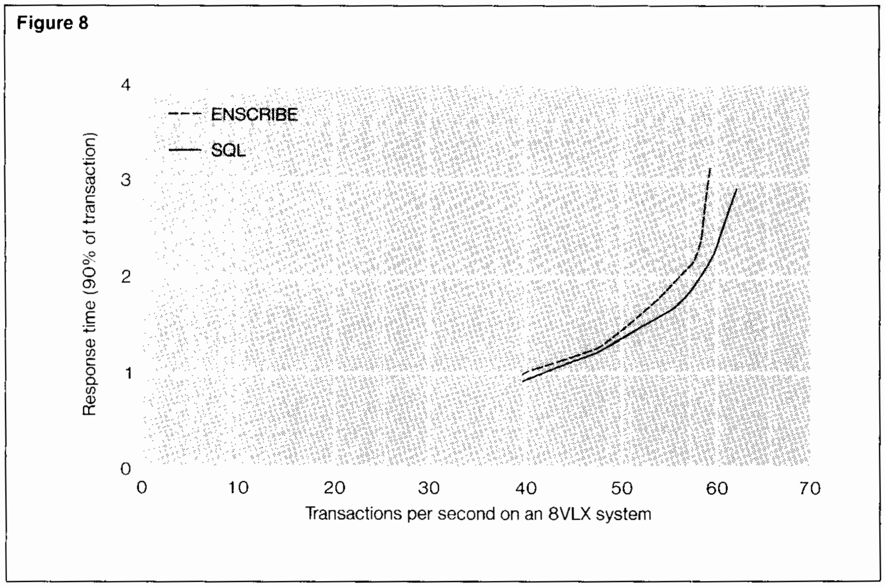


Figure 8.
The response-time curves for NonStop SQL and Tandem's record-at-a-time ENSCRIBE system running on an eight-processor VLX system. Notice that NonStop SQL has slightly better performance than ENSCRIBE.

The hardware configuration is summarized in Table 1. Assembling this hardware was difficult. Because the VLX had a backlog of orders, the equipment was only available for a limited time. The hardware had to be assembled, benchmarked, and disassembled within a 50-day time frame. Fortunately, the equipment was installed on schedule, there were no hardware errors during the benchmark, and no critical software errors were discovered in the NonStop SQL product itself. The only hardware problem was a double power failure at the facility early in the benchmark.

The Experiments

The benchmark measured the ability to grow a NonStop VLX system from 8 VLX processors to 32 processors using the FOX fiber-optic ring. In addition, the benchmark demonstrated that NonStop SQL runs on Tandem's low-end EXT10 system attached to the VLX processors via a 9.6-Kbit line. The EXT10 had a proportionate part of the database and sent and received distributed transactions (15%).

First, the throughput of a single 8VLX system was measured for both ENSCRIBE and SQL. Then, a pair of 8VLX systems were connected via FOX and their performance measured for SQL. Finally, the four-node, 32-processor, FOX-connected network with attached EXT10 was measured. Because ENSCRIBE is limited to 16 partitions per file, the last experiment was done only for NonStop SQL.

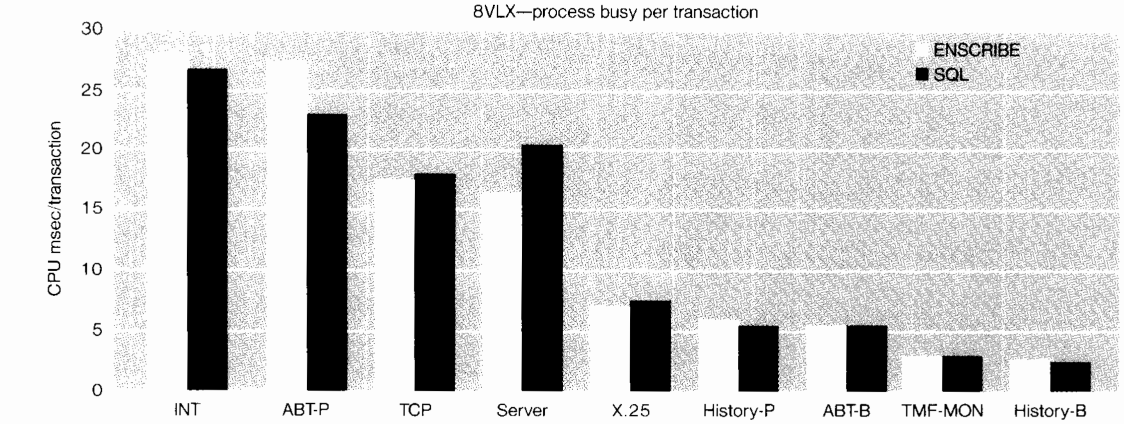
Figure 8 shows the response-time curves for the SQL and ENSCRIBE tests on an eight-processor VLX system. The curves show that, for the same response time, SQL gives slightly better throughput and, similarly, that SQL gives better response time for a fixed throughput.

Each transaction generated one physical read and two physical writes of the account file (one mirrored write). Disk reads and writes of the account, branch, and history files are amortized over many transactions. In addition, each transaction contributes about 0.4 physical log I/O per transaction, because group commit batches about five transactions per audit flush.[1] A detailed breakdown of the CPU utilization by function is shown in Figure 9.

These experiments were repeated for 16 VLX (16VLX) processors. Then the SQL experiments were repeated for 32 VLX processors. (Remember that ENSCRIBE is limited to 16 partitions.) The response times were plotted and are documented in the Auditor's Report (Sawyer, 1987). The resulting throughput curves are shown in Figure 10.

[1] *Group commit* means grouping and consequent amortization of the commit cost of the group of transactions. (This is also called "piggybacking," or "boxcaring.") Even though the commit records are grouped, they are still atomic in nature, i.e., separable by transaction.

Figure 9



These experiments were audited by The Codd and Date Consulting Group,[2] which verified that:

- The transaction was correctly implemented.
- The database was sized properly.
- 15% of the transactions were interbranch.
- Transactions were protected by a dual undo/redo log.
- The response time was measured correctly.
- The measured response-time curves matched the system.
- SQL and ENSCRIBE had comparable performance.

Based on early measurements, the system was expected to perform about 8 tps per CPU and have linear growth from 8 to 32 CPUs, i.e., the 32-processor system would handle approximately 256 tps. In fact, the 8VLX system did 7.2 tps per VLX CPU, and there was a 10% dropoff as the system was scaled to 32 processors. This is shown in Figure 11. The dropoff is due to the increased cost of network distributed transactions. When transactions do work at multiple nodes, they cost extra instructions and I/O traffic. This extra cost implies a reduction in throughput overall. In spite of this, the tps curves are linear; however, the slope is 0.9 rather than 1.

Figure 10

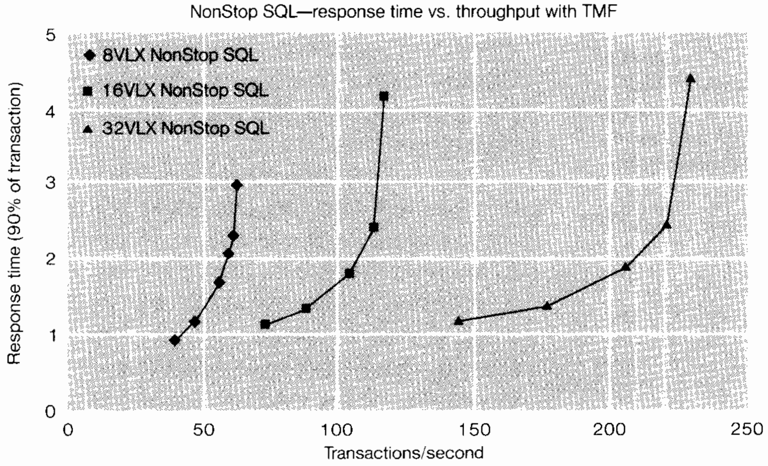


Figure 9.

A detailed breakdown of CPU time spent by each component of the debit-credit transaction on an 8VLX system showing both the ENSCRIBE and SQL costs. INT signifies interrupt handling and the message system. ABT refers to the account, branch, and teller disk servers. Both the primary ABT and backup (process-pair) disk processes are separated. The TCP does presentation services and terminal handling. X.25 does the physical line handling for the input and output messages. The disk server storing the History table is shown separately. Lastly, the transaction commit coordinator is represented by TMF-MON.

Figure 10.

The tps rating of VLX processors for NonStop SQL. At a 2-second (or less) response time for 90% of the transactions, the system was running at a sustained rate of 208 tps.

[2] Founded by E.F. Codd, the originator of the relational model, and C.J. Date, a leading author and lecturer on relational technology, The Codd and Date Consulting Group is considered the ultimate source for relational product education, evaluation, and consulting.

Figure 11

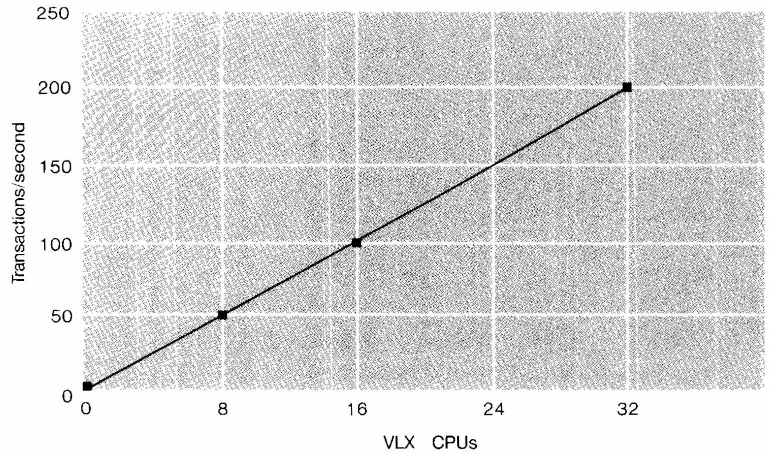


Figure 11.
The tps rating of NonStop SQL running the debit-credit benchmark on VLX processors. On 32 VLX processors the system processed 208 tps.

In addition, NonStop SQL uses fewer CPU instructions than ENSCRIBE in this benchmark. Consequently, NonStop SQL has a slightly higher transaction throughput. Ignoring response time, the peak NonStop SQL throughput was 229 tps.

Given this linear throughput vs. hardware, it is possible to quote the price/performance of the system in terms of the price/performance of a single processor. The five-year cost per transaction is approximately the same for a small system as for a large one. The dominant issue is which processor line the user selects. Generally, newer systems are less expensive.[3]

Why Is NonStop SQL So Fast and Powerful?

Historically, SQL has been confined to information center environments and to low-end systems where programmer productivity is more important than system performance. In these environments, the power of the SQL language and the relational model compensated for the lackluster performance of most relational systems.

NonStop SQL is the first SQL system to offer the high performance necessary for online transaction processing. As demonstrated by the benchmarks, it has performance comparable to Tandem's record-at-a-time ENSCRIBE system. In addition, NonStop SQL is a distributed relational system; other systems do not offer full-function distributed data or distributed execution.

There are several reasons for the success of this benchmark. First, NonStop SQL benefited from the experience of its predecessors. The developers had collectively worked on System R, SQL/DS, DB2, R*, IDM, ENCOMPASS™, Esvel, Wang VS, and several other systems.

A second benefit was the close cooperation between the Database g¯oup and the Operating System and Languages group. Tandem is a transaction processing company; the operating system is geared for efficient processing of distributed transactions. NonStop SQL exploits the transaction mechanism of the GUARDIAN 90XF system code to easily and efficiently get transaction-protected distributed execution based on remote procedure calls.

Perhaps the most significant advantage of NonStop SQL is the SQL language itself, especially when contrasting the record-at-a-time interface of ENSCRIBE, DL/1, or DBTG with the set-oriented interface of SQL. Figure 12a shows sample UPDATE statements in the debit-credit transaction.

In the case of ENSCRIBE, a message is sent to the disk process to get the designated branch record. The record is then returned to the program, which examines it, alters it, and then returns it to the database. The result is actually three messages and a lot of data movement. (See Figure 12b.)

NonStop SQL sends a single message to the disk process requesting an update of the appropriate account by the appropriate amount. The disk process applies this update and returns a status message to the caller. The result is half as many messages and one-quarter the number of message bytes.

In general, NonStop SQL subcontracts single variable queries to remote servers. This allows the NonStop SQL disk process to act as a database machine that performs updates and deletes and filters data, returning only the desired rows and columns of a table.

The examples shown in Figure 12 illustrate the synergy between SQL and distributed database systems. It has long been felt that SQL would be a good basis for distributed database

[3] More detailed pricing information is available in *The NonStop SQL Benchmark Workbook.*

Figure 12

(a)

```
COBOL-ENSCRIBE                                     NonStop SQL
START branch KEY IS EQUAL TO branch_number
READ branch RECORD with LOCK                       UPDATE branch
ADD delta TO balance OF branch                      SET     balance = balance + :delta
REWRITE branch                                      WHERE number = :branch_number
                                                    AND     balance + :delta > =0
```

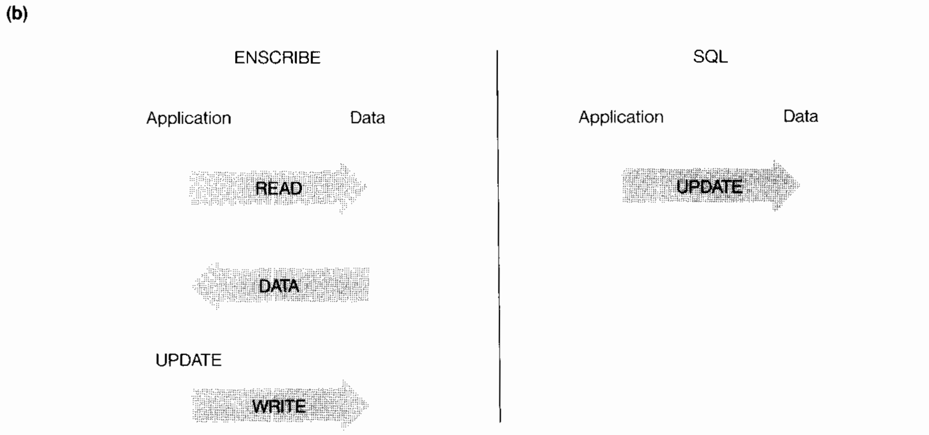


Figure 12.

(a) Comparison of COBOL record-at-a-time updates, which send a read followed by a write, and the SQL update, which sends a single update message. This message savings helps explain the performance advantage of SQL compared to ENSCRIBE.
(b) SQL can update or filter data at the source, whereas record-at-a-time interfaces must fetch the data to the application program. In this way, SQL saves messages and CPU cycles as compared to a conventional data management system.

systems. In every benchmark the message savings of SQL have compensated for the extra work the system must perform in order to implement the more complex semantics of the SQL language. For example, NonStop SQL is nearly twice as fast as ENFORM™, Tandem's report writer for ENSCRIBE files, on the Wisconsin Benchmark (Bitton, 1983). Most of the increased speed is due to the close integration of NonStop SQL with the operating system.

Even in the best circumstances, a system might bottleneck on certain resources. For example, most transaction processing systems bottleneck at 30 tps because they have not implemented group commit, which batches commit records (Gawlick, 1985). The fact that NonStop SQL was benchmarked using 32 processors at over 200 tps, without any bottlenecks, suggests that there are no obvious bottlenecks in the system.

Conclusion

Traditionally, relational systems have had a reputation for poor performance. NonStop SQL, however, performs comparably to traditional, record-at-a-time nonrelational systems. Running over 200 tps on a 32-processor VLX complex and an EXT10 demonstrates that NonStop SQL is functional, distributed, and scaleable. In addition, it has good price/performance and flat price/performance from the low end to the high end. There are no performance limits and no performance penalty for NonStop SQL.

References

Anon, et al. 1985. A Measure of Transaction Processing Power. *Datamation*. Vol. 31, No. 7.

Bitton, D., et al. 1983. Benchmarking Database Systems: A Systemic Approach. Proc. 9th VLDB. IEEE Press.

The NonStop SQL Benchmark Workbook. Part no. 84160. Tandem Computers Incorporated. (Available through Tandem Sales Representatives.)

Gawlick, D. 1985. Processing Hot Spots in High Performance Systems. IEEE COMPCON '85.

The Tandem Performance Group prepared this article: Dan Adachi, Nahn Chu, Frank Clugage, Jim Enright, Ray Glasstone, Jim Gray, Gerhard Huff, Jeff LaPlante, Jack Mauger, Frank O'Donnell, Harald Sammer, Praful Shah, Scott Sitler, and Thomas Wilkens.

Introduction to MULTILAN

Tandem's MULTILAN™, a set of hardware and software products, extends the benefits of Tandem systems to PC users by allowing them to link their NETBIOS-compatible local area networks (LANs) to a Tandem system. The PCs may be connected to a variety of LANs, which can then be linked to any Tandem NonStop system, as long as the LANs adhere to the standards described later in this article.

This article describes LANs, LAN standards, MULTILAN, and various application uses of MULTILAN. For more specific information on the MULTILAN file server and MULTILAN programming, read the accompanying articles, "Overview of the MULTILAN Server" and "Using the MULTILAN Application Interfaces."

LANs

A LAN is a data communications system that connects computing devices within a relatively small geographic area—in one building or among several buildings in an office complex or campus—to share resources. It does this at high data rates (Mbits per second).

One result of connecting multiple terminals and PCs to a host is the physical congestion of wire. A LAN replaces all the dedicated wiring, both eliminating the congestion and reducing costs by allowing users to communicate with and share computers and peripheral devices.

There are three general types of LANs:

- Terminal LANs that connect large numbers of interactive terminals to a host computer.
- Back-end LANs that connect multiple mainframe computers.
- PC LANs that connect personal computers and peripheral devices.

LAN Standards

Prior to 1984 there were no industry-wide standards for LANs, and LAN vendors tended to follow their own rules. The lack of consistent standards created many problems, the most serious being that software developers were hindered from writing network LAN applications.

In 1984 IBM announced PC Network to interconnect PCs over a LAN in departments and offices. This was the beginning of the PC explosion and it is estimated that by 1990 80% of PCs will be connected to LANs. With the introduction of IBM's PC Network, for the first time both resource sharing and personal communication were made more attractive to the PC user.

The announcement of the PC Network LAN included both hardware and software, the software being *DOS 3.x* and the *PC LAN Program* (formerly the PC Network Program).[1] The hardware delivered with PC Network includes such components as cables, translator unit, and network adapter cards. (The network adapter card is inserted into each PC, thereby connecting the PC to the LAN.) The major features of the PC LAN Program include:

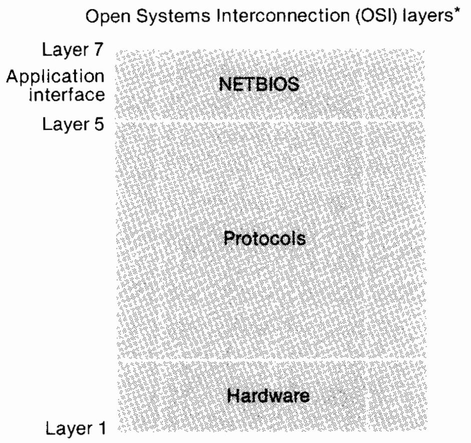
- A NETBIOS (Network Basic Input Output System) application interface.
- The redirector.
- The server.
- Server Message Block protocols.

DOS 3.x. DOS is a disk-based operating system used for controlling resources on a micro or personal computer. There are various versions of DOS available, including but not limited to 2.0, 2.1, 3.1, and 3.2.[2] There can be major differences in the functions provided by various versions of DOS; some of the differences relevant to LANs are discussed next.

Applications written under DOS 2.x with a PC LAN vendor's product could access files on remote disks in an exclusive open mode. The software controlling the remote disk was referred to as a "disk server." The disk server allowed several PCs to share a common "virtual" disk but controlled access to the entire disk. However, the disk server's I/O requests were on a block or sector level and could not control access on a file level.

Resource sharing was made easier for the PC user with the introduction of DOS 3.x and the implementation of a file server. A *file server* is software running on a PC which processes file-access requests from several PCs on a LAN. It is a central point of control for shared peripherals such as disks and printers. The file server is able to maintain a high level of data integrity if applications accessing shared disk files make use of open modes and file and byte-range locking.

Figure 1



*These layers correspond closely to the OSI Reference Model. The OSI Reference Model is a model developed by the International Standards Organization (ISO).

Figure 1.
NETBIOS standard application-programming interface. NETBIOS is independent of the underlying protocols.

A function of DOS 3.x lets PCs share files on a network. This sharing is done in two different modes: the access mode and the sharing mode. The access mode allows a network application to open a file and specify how that file will be used: read only, write only, or read and write. The sharing mode allows a network application to specify whether or not the open file can be read or written by other network applications.

With database activity, more than one application needs to modify the same file at the same time. Therefore, network file-access control for a single file must extend down to the record level. Record locking is only available in DOS 3.x and allows controlled access to different records in a file by multiple applications.

NETBIOS. Since its introduction by IBM, NETBIOS has been the de facto standard for a session-level interface (layer 5) to network applications (see Figure 1). NETBIOS is a "peer-to-peer" standard application-programming interface so there is no concept of a "host." It therefore allows programs written using this interface to cooperate. It is important to understand that NETBIOS is an interface and not a protocol and therefore is independent of the underlying LAN protocols.

[1] With IBM's introduction of Token-Ring, PC Network was replaced with the PC LAN Program to demonstrate support for both the PC Network LAN and the Token-Ring LAN. Functionally, they are the same.

Microsoft Network, referred to as MS-NET, is a product of Microsoft Corporation. MS-NET is used by some LAN vendors to provide services similar to those provided by the PC LAN Program for IBM networks.

[2] For convenience, the terms *DOS 2.x* and *DOS 3.x* are used here to mean any version of DOS 2 and DOS 3, respectively. It refers equally to IBM PC-DOS and Microsoft's MS-DOS.

A session is a logical, two-way, point-to-point connection between two applications. It provides the ability for data transfer for a period of time. NETBIOS allows the establishment of a session between named entities on the network. A name on the network is just a logical end point for communications. NETBIOS provides a means for an application to introduce its name to the network.

Applications can exchange information by way of their names via NETBIOS through sessions or datagrams. A session allows two names to exchange data reliably by sending and receiving messages. A datagram is an unreliable transfer of data from one name to another name or to a group name on the network. IBM's initial implementation of NETBIOS had a limitation of 16 names and 32 sessions on each network adapter card.

In short, NETBIOS allows applications to connect to a network. Because of this standard interface, network applications can now be ported from one network to another regardless of the underlying LAN protocols.

Redirectors and Server Message Block Protocol. The *redirector* is a software component of the PC LAN Program that uses a LAN to link the shared resources managed by the file server to PCs on the LAN. The redirector allows PC programs that use DOS 3.x file-system calls to transparently have those calls rerouted, across a network, to a file server that actually executes those calls. It routes the call to the appropriate file server, passing the information in the file request via messages. The messages exchanged between the redirector and the file server follow a specific protocol, which is referred to as the Server Message Block (SMB). The SMB protocol consists of a set of rules that govern the sending of data structures that represent file-access requests over a NETBIOS session.

When an application wishes to perform a FILE OPEN request via DOS, DOS examines the request to determine if it refers to a network device. If the request is destined for a network device, DOS passes the request to the redirector. The redirector packages the request into an SMB message and sends the request using a network message to the file server over a NETBIOS session.

The message sent by the redirector travels over the network to the adapter card in the PC running the file server. The message is received by the NETBIOS component in the file-server machine and passed to the file-server software, which interprets the SMB protocol and performs the underlying request (e.g., create directory, open file, etc.).

MULTILAN

The MULTILAN product design is based on the standards described previously. Not itself a LAN, MULTILAN allows users to protect their LAN investment by connecting third-party LAN hardware and software that is NETBIOS-compatible to Tandem systems. In addition, MULTILAN allows users to choose a LAN vendor without having the choice dictated by Tandem.

The MULTILAN Access Method (MLAM) is based on the NETBIOS application-programming interface and the MULTILAN file server. The server requires DOS 3.x and the redirector, which supports the SMB protocol.

Most large corporations have different types of LANs installed to support different areas of the business. For example, on the factory floor, Ethernet may be required to support the manufacturing of a product. In the administrative areas of the corporation, Token-Ring may be required because the building is already wired to IBM cable standards. Both of these installations, however, need to share the same data.

Multiple LAN types mean multiple protocols. This represents the technical challenge of interfacing multiple technologies while still providing for the sharing of data and resources.

NETBIOS is a standard application interface independent of the protocol of a LAN. Applications written to the NETBIOS standard allow these applications to run on different LAN types. With this standard and other features, MULTILAN adds value to your LAN that most LAN vendors cannot provide.

MULTILAN Hardware

MULTILAN provides fault tolerance and modular expandability consistent with the Tandem architecture philosophy; it can be expanded to support an ever-increasing PC population.

There are four main hardware components required for a MULTILAN installation (see Figure 2):

- A 5600 controller.
- An Ethernet transceiver and associated cable.
- A 10-Mbit-per-second thin Ethernet cable.
- A MULTILAN attachment device (MLAD).

The 5600 controller is a dual-ported controller that connects to two Tandem processors via two I/O channels for fault tolerance. The 5600 uses dual lock-stepped Motorola 68000 microprocessors and dual direct memory access (DMA) chips for data integrity and an Intel 82586 LAN coprocessor for Ethernet media access control. The Ethernet transceiver provides the interface between the 5600 and the thin Ethernet cable. The thin Ethernet cable is then used to connect the transceiver to the MLAD. Multiple controllers can be attached to the same or different LANs (see Figure 3), and a LAN can be attached to more than one Tandem system.

Figure 2

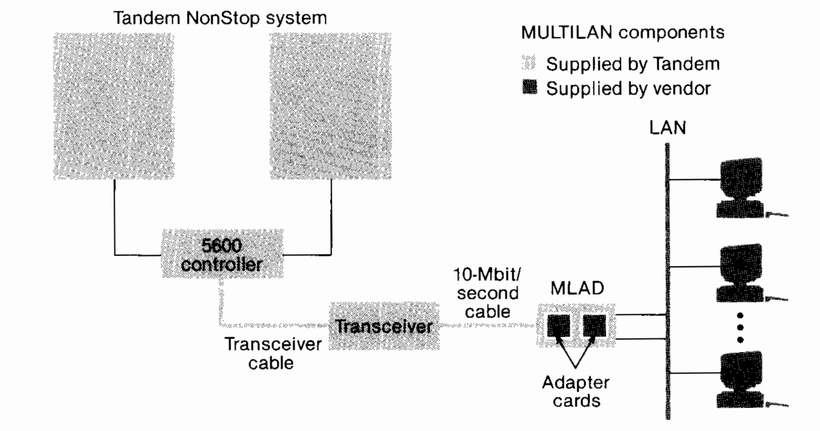


Figure 3

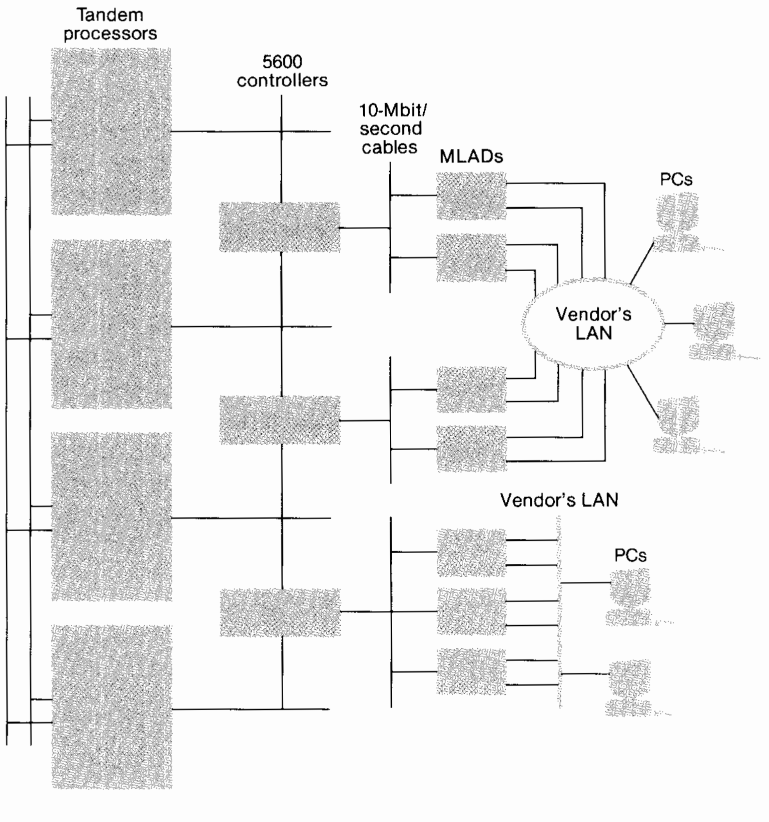
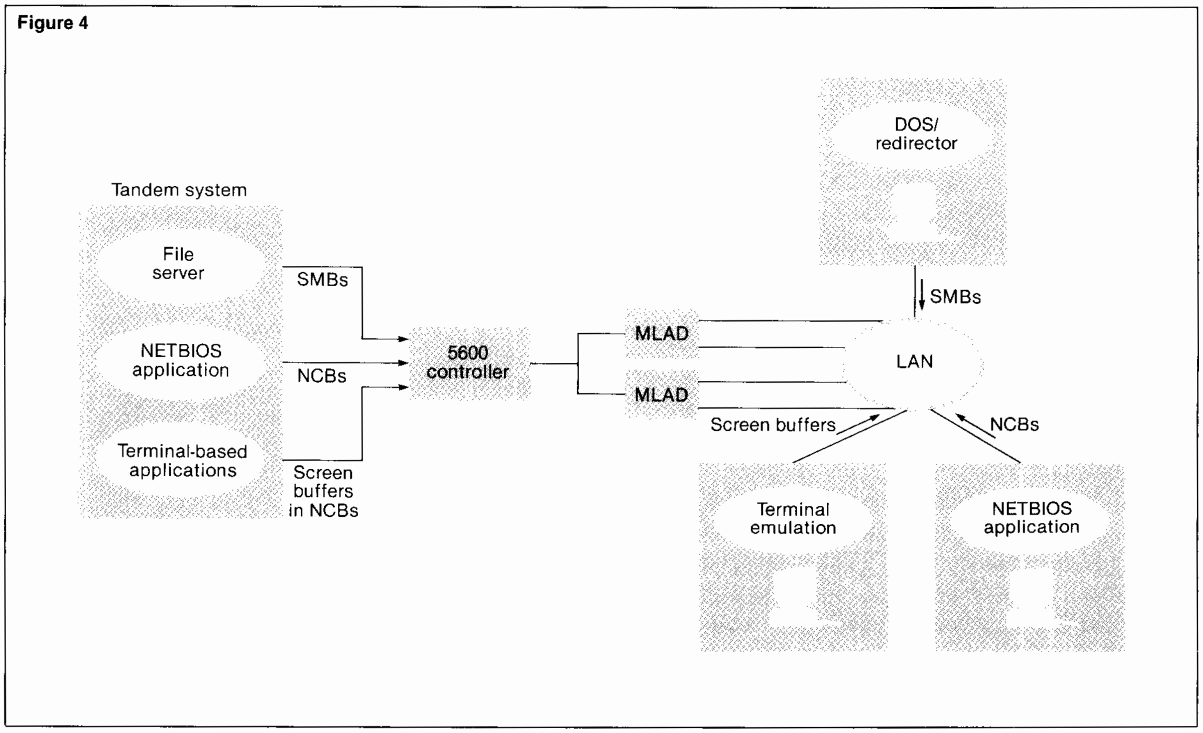


Figure 2.
Main MULTILAN hardware components.

Figure 3.
MULTILAN support for multiple LAN types. Multiple controllers can be attached to the same or different LANs.

Figure 4.

MULTILAN functions. MULTILAN supports a file server, distributed NETBIOS-based applications, and terminal-based applications.



The MLAD is a Tandem 6AX™ workstation containing an Ethernet adapter card used to connect to the 10-Mbit-per-second cable. Up to eight MLADs can be used with each 5600 controller for higher throughput, capacity, or fault tolerance. The 10-Mbit-per-second cable is a standard RG58 (A/U or C/U) coaxial cable that links the 5600 controller to the MLAD(s).

The installation of the MULTILAN product is similar to the installation of other Tandem hardware. The Tandem customer engineer installs the 5600 controller in the Tandem system and connects the cables and auxiliary equipment from the 5600 to the MLAD(s).

The system manager generates a new GUARDIAN 90™ operating system that contains system software to manage the 5600 controller. The LAN vendor's adapter cards are then installed in the MLAD. Up to two LAN adapter cards can be installed in one MLAD, depending on the LAN type. The number of MLADs and 5600 controllers required depends on the number of sessions and/or names handled by the LAN vendor's adapter cards and the number of PC users on the LAN. Finally, DOS 3.x, the LAN vendor's NETBIOS, and Tandem's MLAD software are loaded into the MLAD from a floppy disk.

Upon completion of the installation, MULTILAN transforms the Tandem system into a "PC node" on the LAN. Any messages destined for the Tandem system are intercepted by the MLAD and sent across the 10-Mbit-per-second link to the 5600 controller. The 5600 controller passes the messages to the MLAM.

MULTILAN Software

MULTILAN consists of a comprehensive set of software products that allow LAN-connected PCs to interact with Tandem host applications (see Figure 4). The MULTILAN software is composed of host- and PC-resident software. (See Figure 5 for the process-structure.) The major components are the following:

- MLAM.
- PC terminal emulator (PCT or EM6530PC).
- MULTILAN Server Software (MLSRV).
- MULTILAN Resource Manager (MLRES).
- Resource Manager for the PC (RM/PC).
- Subsystem Control Facility (SCF).
- Subsystem Control Point (SCP).
- MULTILAN Manager (MLMAN).
- Tandem Maintenance and Diagnostic Subsystem (TMDS™).

MLAM. MLAM is host-resident software that provides the lower-level protocols for passing data from the Tandem system to the MLAD via the 5600 controller. The MLAM is the I/O process that runs in both a Tandem processor and in the 5600 controller. MLAM comprises two primary interfaces, WS6530 (a 6530 emulation interface) and WSPTP (a process-to-process interface). An ancillary component to WS6530 and WSPTP, CONFIG provides a configuration and management interface to MLMAN.

MLAM and WS6530 with PCT. The WS6530 host-resident interface, supplied by MLAM, permits a high-level access from LAN-connected workstations to the Tandem system (see Figure 6). It allows workstation users to access 6530 terminal-oriented applications such as block-mode PATHWAY and PS MAIL™. No changes are required in the Tandem block-mode applications when using WS6530. WS6530 uses NETBIOS internally; however, the application designer and programmer are spared the details of NETBIOS usage. The WS6530 interface to workstations is used by the PC-resident EM6530PC emulator software.

Figure 5

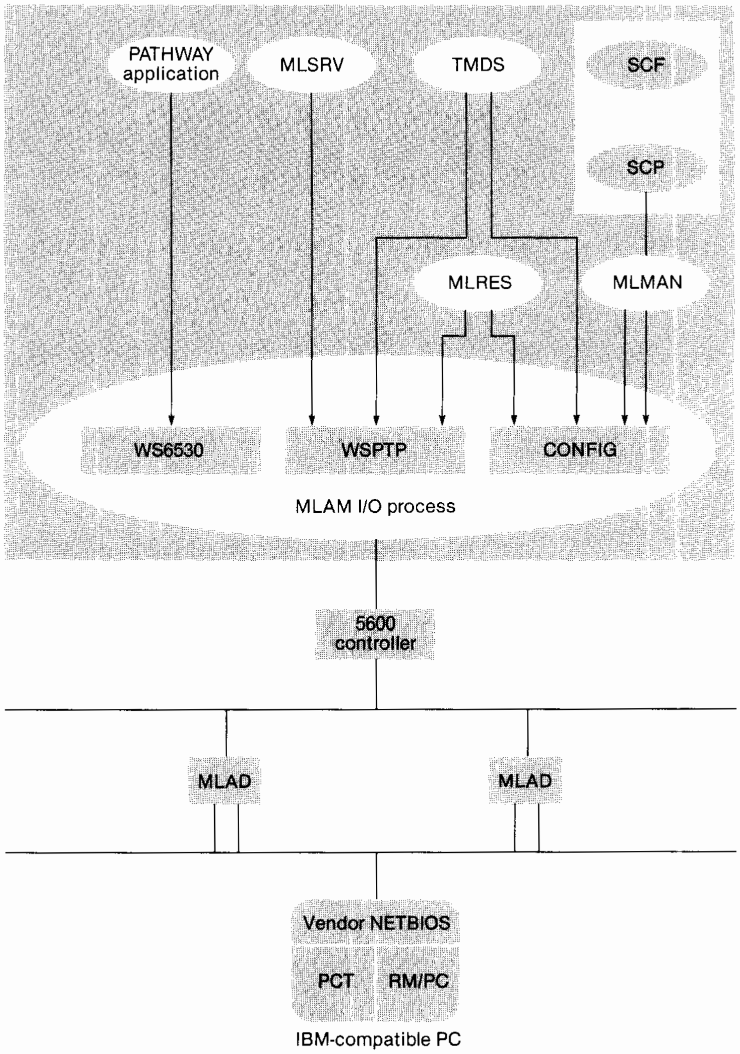


Figure 5.
MULTILAN software overview. A system manager's view of MULTILAN.

Workstations requiring access to Tandem block-mode applications must be running the EM6530PC emulation program. To run EM6530PC software, the following components are required: PCT.INI, LAN6530.SYS, KERNEL.SYS, and RMPCDRVR.SYS.

Figure 6

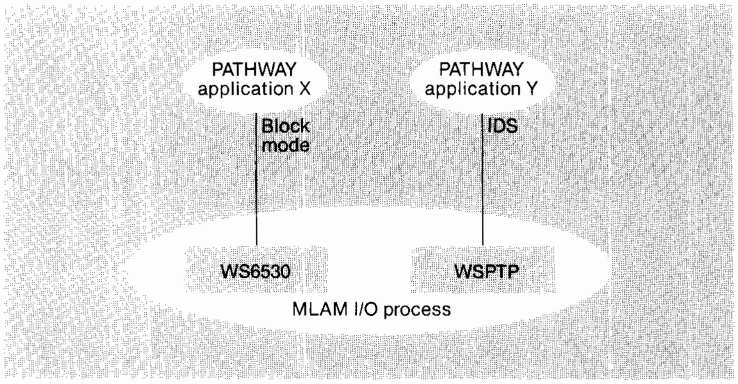


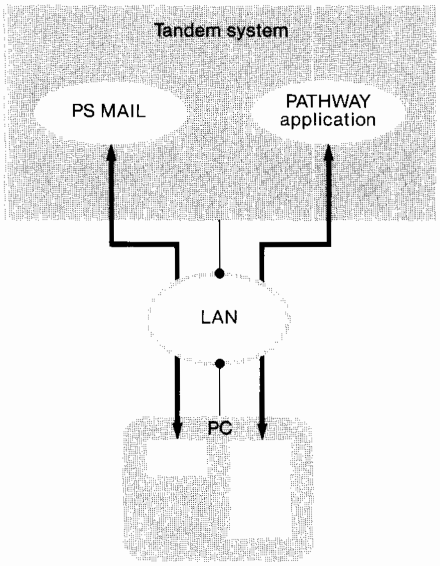
Figure 6.

A programmer's view of MULTILAN.

Figure 7.

MULTILAN terminal emulation with windowing software. Terminal emulation with a windowing package allows simultaneous access from a workstation to multiple Tandem applications.

Figure 7



Multiple copies of the EM6530PC software can be run on a workstation in combination with a windowing package. This allows simultaneous access from a single workstation to multiple Tandem host applications executing on one system (see Figure 7). As an added feature, applications run on remote Tandem systems can be accessed by workstations connected to a local system across the Tandem EXPAND™ network.

MLAM and WSPTP. The WSPTP interface supplied by MLAM provides a NETBIOS-level (and NETBIOS-like) interface between a PC application and an application running on the Tandem host (see Figure 6). The WSPTP interface provides the capability to write applications that run partially on PCs and partially on the Tandem host. An application structured in this manner is referred to as a *cooperative processing* application.

Using cooperative processing, applications can be designed so that PC-resident application software processes data manipulation and presentation services, and the Tandem host-resident application software can process database access requests (see Figure 8). PC applications can communicate with each other as well as with the Tandem system through a common NETBIOS interface. By means of this interface, broadcasting of information, such as a stock-update application, allows the updating of PCs without having to uniquely address PCs on the LAN. Designs of this nature can reduce LAN traffic and offload processing from the host to the workstation. As a general rule, response time is improved due to the screens and all related processing being executed locally in the PC.

MULTILAN Server. The MULTILAN Server software (MLSRV) is host-resident software that allows PC users to store and retrieve DOS files on the Tandem system and submit print jobs to the Tandem spooler. The Tandem file server allows PC users to access DOS files as if they were stored on a network file server. The files can reside on a local Tandem system or on any Tandem node in an EXPAND network (see Figure 9).

The MULTILAN file server supports the redirector SMB protocol for communication between the PC and the Tandem system. PC users can make files available to other PC users, thereby providing for resource sharing. MLSRV allows PC users to access GUARDIAN 90 files using standard DOS commands and provides the facilities to transfer DOS and GUARDIAN 90 files to and from the Tandem spooler using standard DOS commands.

MLRES, RM/PC, SCF, SCP, and MLMAN. MLRES is host-resident software that verifies and logs on PC users to the Tandem system. MLRES also dynamically creates Tandem host applications for PCs running in 6530 emulation mode, and it helps to establish a WS6530 session between the PC and the Tandem host application.

The connection between the host 6530 datastream application and the EM6530PC program is achieved through a logical entity called a window. A *window* is a facility used for conducting a NETBIOS session between the EM6530PC program and the Tandem application. Sessions between a PC and the Tandem system can be established in either a dynamic or static mode.

A dynamic session is created only when an application is dynamically invoked by an EM6530PC program. This program, RM/PC, and MLRES all interact to create the application, session, and associated dynamic window.

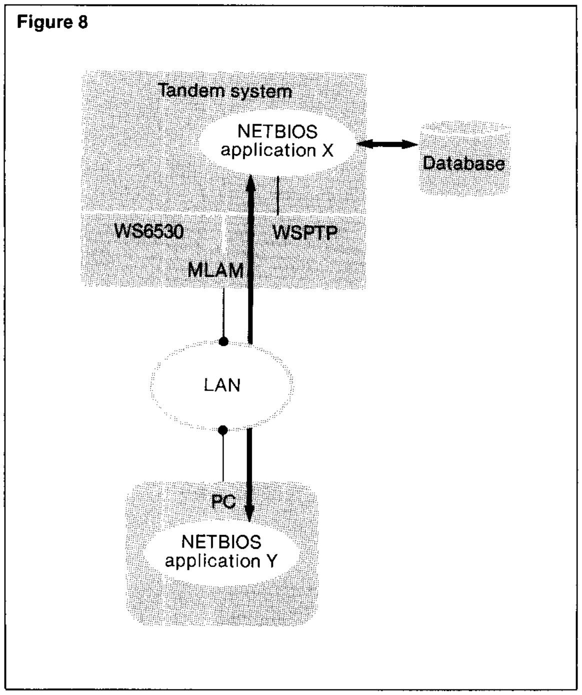


Figure 8.
A MULTILAN cooperative-processing application. PC applications and Tandem applications communicate through a common NETBIOS interface.

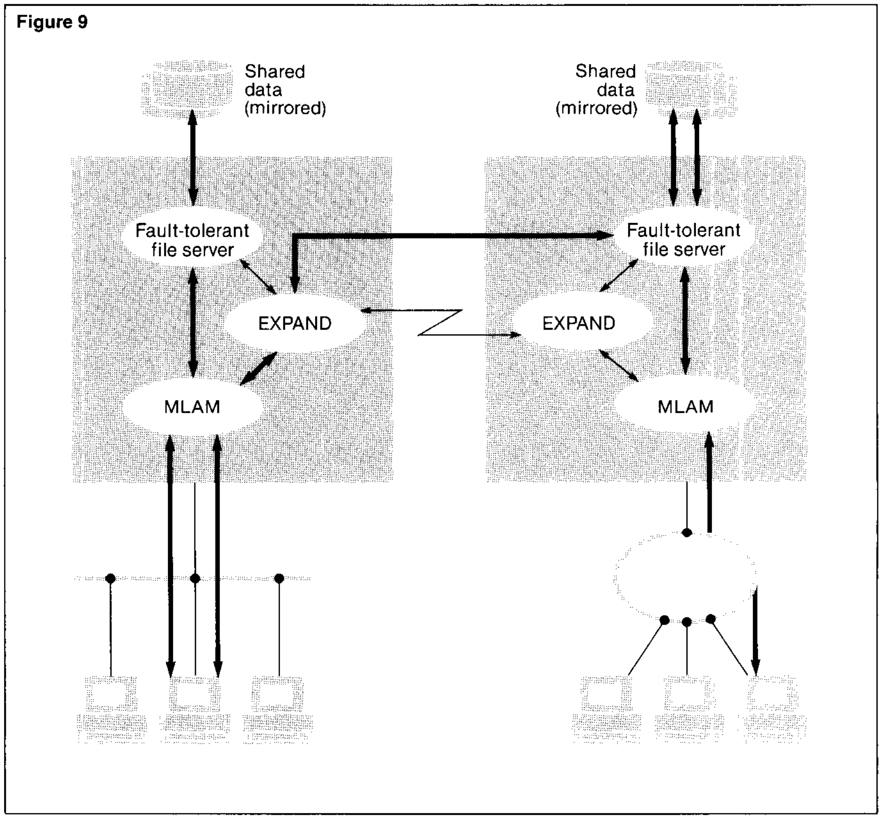


Figure 9.
MULTILAN distributed file-server access. PC applications can transparently share data on a local file server or across a Tandem network.

A static window is started and stopped by the system manager. The system manager preconfigures a static window on behalf of the PC user into the MLAM using an operator interface called the Subsystem Control Facility (SCF). SCF then uses another component, the Subsystem Control Point (SCP). The static session is created when, upon invocation, the EM6530PC program calls that static window.

SCF and SCP are host-resident software that make up a general subsystem control facility for configuring and managing various entities in a network. (The SCF and SCP components are part of Tandem's Distributed System Management product offering.) SCF passes the request from the system manager through SCP to the MLMAN for processing. MLMAN is host-resident software that provides configuration and management capabilities specific to the MULTILAN environment.

TMDS. TMDS is host-resident software that is a general maintenance and diagnostic subsystem for a variety of Tandem hardware and software products, of which MULTILAN is one.

Conclusion

MULTILAN is based on industry standards that provide access to the widest range of PC LANs. In addition to delivering significant facilities to PC users on LANs, MULTILAN helps users take advantage of advanced applications while continuing to build on Tandem's traditional strength in on-line transaction processing.

References

IBM PC Network Technical Reference Manual. International Business Machines Corporation. Part no. 6322916.

IBM PC LAN Program Manual. International Business Machines Corporation. Part no. 6139747.

IBM Personal Computer Seminar Proceedings. May 1985. International Business Machines Corporation.

LAN Software Report. 1986. Novell, Inc.

Lewis, J. 1986. DOS 3.1. *LAN Interface Magazine.*

______. 1986. NETBIOS. *LAN Interface Magazine.*

MULTILAN Manager's Guide. Tandem Computers Incorporated. Part no. 82789.

MULTILAN Programmer's Guide. Tandem Computers Incorporated. Part no. 82488.

Steen, R. 1986. IBM Local Area Network Directions: The New LAN Era. LocalNet Seminar Proceedings. International Business Machines Corporation.

Acknowledgments
The author would like to thank Mike Berg, Bart Grantham, Claire Seals, and Jeri Edwards for providing technical review of this article. A special thanks to Alan Rowe, Len Fishler, Peter DeSouza, and Mark Anderton for their technical reviews, many suggestions, and support.

Anne Coyle joined Tandem in New York as an account analyst in July 1983. In 1985 she moved to the San Francisco office where she installed and supported one of the first MULTILAN beta sites. Most recently Anne helped design a LAN solution using MULTILAN for a major Tandem customer. Currently, she is a district systems manager for the Finance and Utilities District.

Overview of the MULTILAN Server

A *file server* is a program that allows one "server" computer to store files on behalf of other computers, most often PCs. The PCs and the server normally communicate over a local area network (LAN).

This article describes the design requirements and implementation of the MULTILAN file server, MLSRV. MULTILAN LAN integration products, as described in the accompanying article, "Introduction to MULTILAN," allow PCs to be connected to Tandem systems over a variety of LANs. MLSRV allows PCs attached to those LANs to keep their files on the Tandem system and access them using DOS commands.[1]

PC Network

IBM's introduction of PC Network in 1984 helped to gain the acceptability of LANs and file servers within the business community. PC Network is composed of many pieces, the heart of which is a broadband LAN and the PC adapter cards that attach PCs to it. Also included is a message-level interface for LAN communication, NETBIOS (Network Basic Input Output System).

PC Network software has two components: DOS 3.x and the PC LAN Program (formerly the PC Network Program). DOS 3.x provides network-related extensions to DOS 2.x, such as byte-range file locking and the various modes in which multiple PCs can open and share files.[2]

The PC LAN Program provides the code to send messages to and from file servers over the LAN via NETBIOS, and to control the operation of the file servers. The program has a number of components, including the server and the redirector.

The *server* allows a PC to share its disks and printers with other PCs. The *redirector* allows PCs to send out I/O requests to a server. The protocol for redirector-server communication is called the *Server Message Block (SMB).*

Sharing a File Server's Resources

Before a file server can be accessed by PCs, it must offer to share its resources. Those resources can be printers or directories with subdirectories under them.

[1] For simplicity, the MULTILAN server is referred to in this article as a "file server." It is more properly called a "file and print server," however, as it also supports the SMB protocol commands for sending spooled output to printers.

[2] For convenience, the terms *DOS 2.x* and *DOS 3.x* are used here to mean any version of DOS 2 and DOS 3, respectively.

The server offers its resources when its operator enters NETBIOS commands such as

NET SHARE money = \finance\accounts,

which allows users access to all the files and subdirectories in the " \accounts" directory, and

NET SHARE printer = PRN [password],

which allows users to use the server's printer, PRN. (See Figure 1.) In the examples, the parameters "money" and "printer," known as *shortnames*, are the logical names of the shared devices. The logical name is the name by which the PCs know the device. Simple password protection, as shown in the second example, is provided on all shared resources.

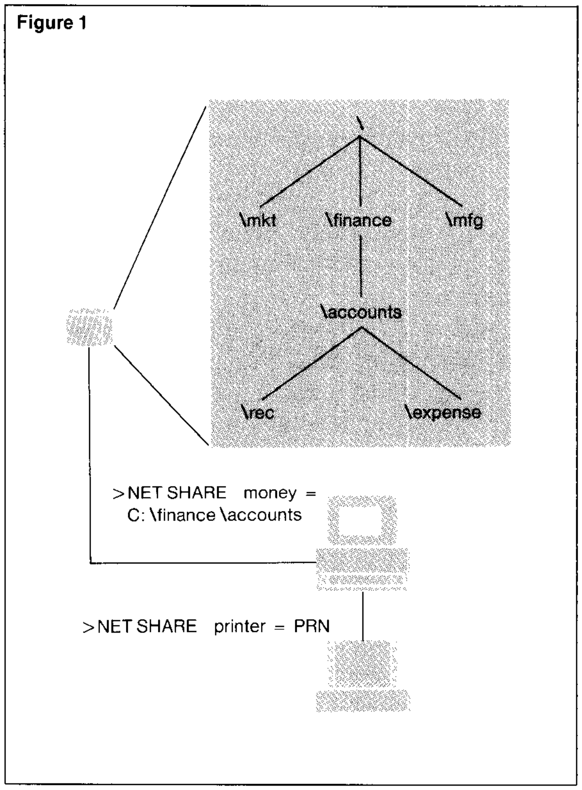


Figure 1.
Configuring a file server to share its resources. These examples use the NET SHARE command to allow users access to all the files and subdirectories in the "\finance\accounts" directory and use of the server's printer, PRN.

Connecting a PC to a File Server

In a PC, the redirector allows a DOS drive (such as N:) to be used as if it were a local drive when it is actually a directory maintained by a remote file server.[3] Similarly, the redirector allows a printer device (such as LPT2) to be used as a local printer when it is actually a spool print job maintained by a server.

The connection to drive N: is made by PC commands such as

NET USE N: \\FSRV \money,

which makes drive N: appear to be a disk with all the subdirectories and files of the server's "accounts" directory, and

NET USE LPT2: \\FSRV \printer [password],

which causes all output sent to the PC's printer, LPT2, to be sent to the server's printer, PRN. In these examples, FSRV is the NETBIOS name of a file server on the LAN, and "money" and "printer" are the shortnames of the resources that the file server shared in the previous examples. The sharing and connection are shown in Figure 2.

On the first connection to a particular file server, the PC LAN Program establishes a NETBIOS *session* between the PC and the file server. The session is a logical path between these endpoints over the LAN. Further NET USEs to the same file-server name are multiplexed over the same session.

In a user PC, the redirector intercepts all DOS I/O calls. If it recognizes a reference to a nonlocal device such as N:, it formats an SMB message and sends it (over the NETBIOS session) to the file server connected to drive N:, as shown in Figure 3. SMB protocol contains commands equivalent to all the operations DOS can perform, such as OPEN, READ, WRITE, CLOSE, MAKE DIRECTORY, and RENAME. If the redirector sees an I/O reference to a local device, it simply passes it to DOS.

[3] From this point on, the name *DOS* refers equally to IBM's PC-DOS and Microsoft's MS-DOS.

File Servers Currently Available

After the introduction of NETBIOS and SMB, many computer vendors began developing file servers. Most provide PC-based file servers, in which a PC is a file-server station on the LAN, and, in most cases, is dedicated solely to file-server activity. (The PC LAN Program allows a PC to be a redirector and server simultaneously; however, this seems undesirable in a production environment, as server code could interfere with application code and performance problems could arise.)

Tandem customers have been using the MULTILAN server since November 1986. Currently, it is the only fault-tolerant file server designed for a mainframe. Its design considerations are discussed next.

File-Server Design

Important file-server design issues include:

- Performance.
- Tools.
- Reliability and availability.
- Connectivity.
- Expandability.
- Cost.

Performance

A file server must support all users with reasonable performance. Internally, it must support many concurrent I/O requests or it will probably perform poorly. CPU performance and disk speed are also important. If CPU power is insufficient, more servers or higher-performance servers must be installed; if disks are slow, more of them must be used to benefit from multiple parallel I/Os.

Tools

File servers that support more than a few users are often controlled by a data processing department rather than by individual users. Tools for file backup and restoration, disk space analysis, backup tape catalogs, performance modeling and prediction, and maintenance and diagnostic functions are important.

Figure 2

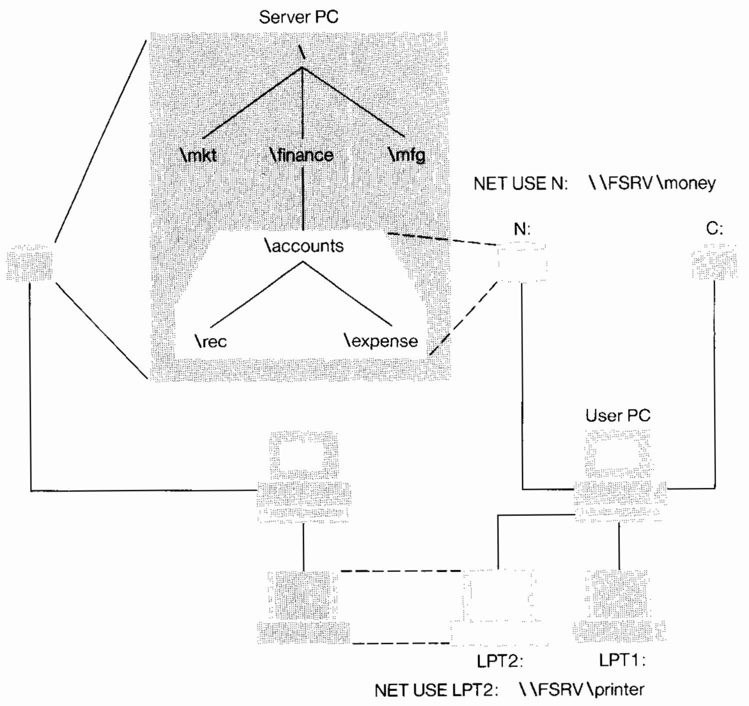


Figure 3

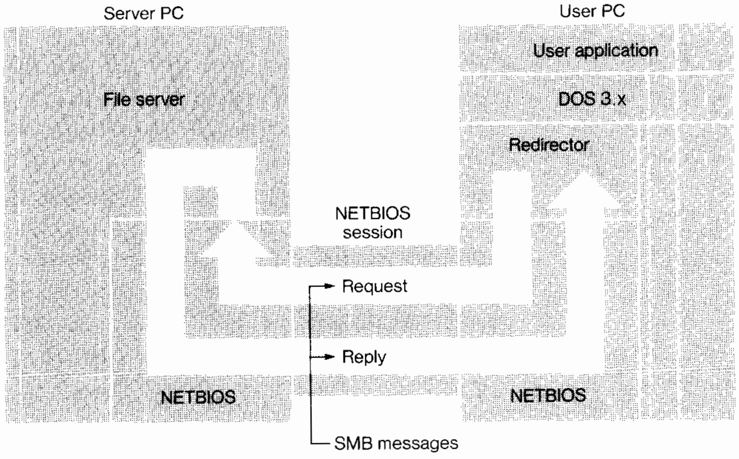


Figure 2.
Sharing and connection between a PC and the file server.

Figure 3.
The redirector intercepts all DOS I/O calls. If it recognizes a reference to a nonlocal device, it formats an SMB message and sends it (over the NETBIOS session) to the file server connected to that device.

The file server must provide such tools, or it must be designed so that the standard operating system tools can be used. Since the staff must be educated to use these tools, it is often easier to allow existing system tools to be used than to develop custom tools and the accompanying manuals and customer training.

Reliability and Availability

These are among the most important issues to be considered. First there is the problem of backing up PC disks. As this is a tedious job, many employees do it only irregularly, causing considerable risk to the company. It is easy to make operational errors during backups also. Errors such as reusing the same floppy for the backup can be just as serious as not backing up the disk at all.

Second, there is the problem of file-server failure. Loss of the file server's data files is far more serious than the loss of a single-user PC, since all users of that file server are affected by the loss. Also, as file servers typically have larger disks than user PCs, backing up their files becomes more important, yet is often clumsier.

A third problem is disk failure. This is normally a hardware problem, but it can also be caused by poor software. For example, if the CPU is rebooted during a critical disk operation (such as a write to a free-space map) and the disk software is not designed to anticipate this, the disk can be corrupted. Many PC users have noted that their disk data structures (e.g., cluster maps and directories) can become corrupted for no obvious reason. Some PC companies even provide a tool that can be used to detect and "fix" disk inconsistencies, since the problem is so common.

Finally, there are CPU, controller, and power failures. Under any of these conditions, the file server is lost and business stops. Indeed, any company using a file server that is not fault-tolerant may well be putting the company at unacceptable risk.

The design and implementation requirements for fault tolerance are not trivial. They include, at the least, duplicate hardware (duplicate or "mirrored" disks, duplicate CPUs, and duplicate power supplies) and sophisticated software able to support duplicate messages, switch to the alternate hardware after a fault, and restore service to repaired units.

Closely allied to fault tolerance is availability. Many users wish to continue to use a file server even under fault conditions. They must be able to add hardware for this, and the file server must be capable of using the extra hardware when the normal configuration is unavailable.

Connectivity

In the simplest configuration, a file server attached to a LAN services requests from PCs on that LAN. What happens if, for some reason, the users have to be on more than one LAN (e.g., if the LAN has reached its maximum user limit or if the users are in different wings, on different floors, in different buildings, or even in different geographical areas)?

A file server's primary purpose is to allow users to share data, and physical problems such as LAN cabling should not interfere with this. PC users in one department should not be isolated from those in another department just because they are in different buildings. Generally, this means that a file server must be able to deal with multiple LAN configurations.

Expandability

Often, the initial configuration of a file server becomes unsuitable as the number of users increases or the workload per user increases. Expandability is therefore a requirement: the file server must be able to deal with increased disk space and CPU power and with additional LANs and controllers.

Cost

A file server can consist of software only (running on standard hardware), or it can be a combination of software and hardware. Its cost varies accordingly. Also, while actual cost is important, more important (though more difficult to measure) is the cost per feature, per user, and per megabyte of disk, and the cost versus risk of using unreliable software and hardware.

MLSRV Implementation

MLSRV is fully compatible with SMB and NETBIOS, implementing all of the SMB commands sent by the redirector. (It is not compatible with file-server protocols that are not based on NETBIOS or do not use SMB protocol, even though they appear to be compatible with DOS 3.x.)

Figure 4

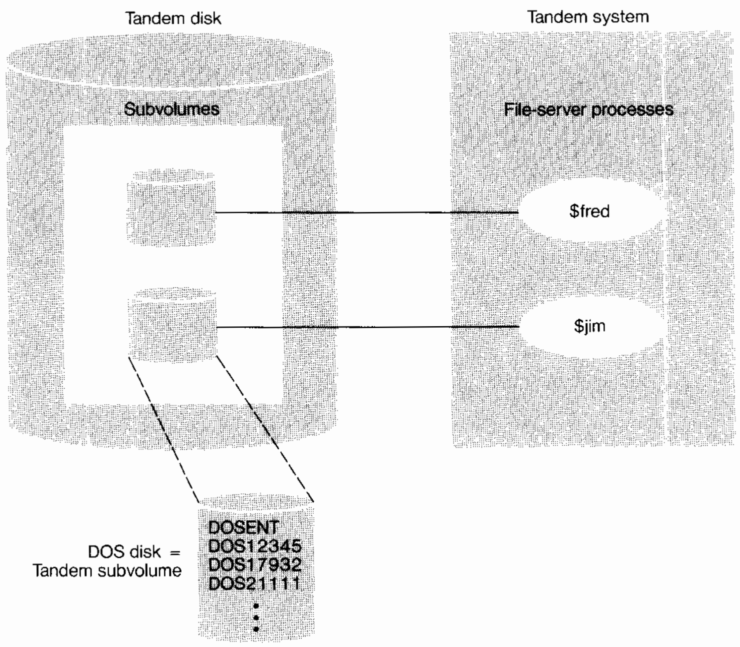


Figure 4.

The DOS files maintained by a file-server process, along with the information to maintain them, are located on a single GUARDIAN subvolume.

Basic Structure

File servers run as processes under Tandem's GUARDIAN 90 operating system. In the PC, the standard DOS 3.x and PC LAN Program are used. No part of MLSRV runs in the PC.

There is no practical limit to the number of file-server processes that can run on a Tandem NonStop system and, thus, no practical limit to the number of DOS disks that can be maintained. (Each PC is limited, by DOS, to accessing 26 disks at one time.)

Each MLSRV process controls what, to PC users, appears to be a single DOS disk containing DOS directories and files. The files are actually GUARDIAN files, their names (in the form DOS*nnnnn*, where *n* is numeric) chosen by MLSRV. MLSRV translates DOS file I/O requests from the user PCs into equivalent operations on the GUARDIAN files. It also maintains a key-sequenced file whose records keep track of DOS directory names and attributes, provide the transformation of the DOS file name to the GUARDIAN file name, and keep track of the various DOS attributes (e.g., read-only and creation date and time).

The DOS files maintained by a file-server process, along with the information to maintain them, are located on a single subvolume, as shown in Figure 4. Each subvolume and, thus, each DOS disk can be as large as a physical disk, or about 450 Mbytes.

Figure 5

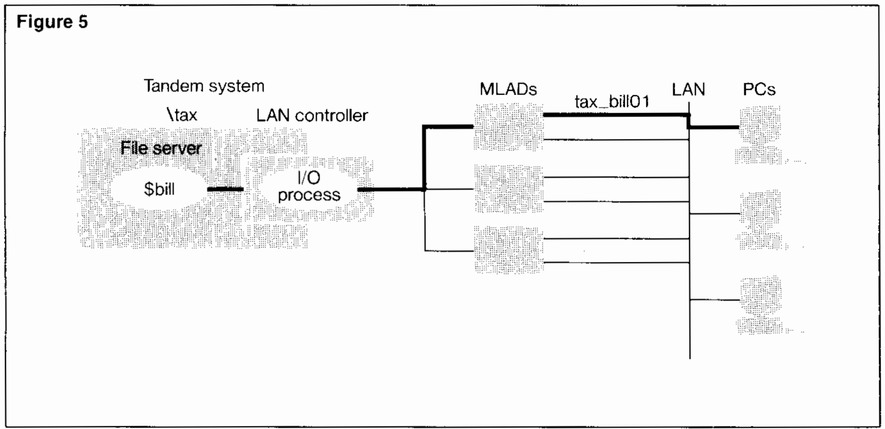


Figure 6

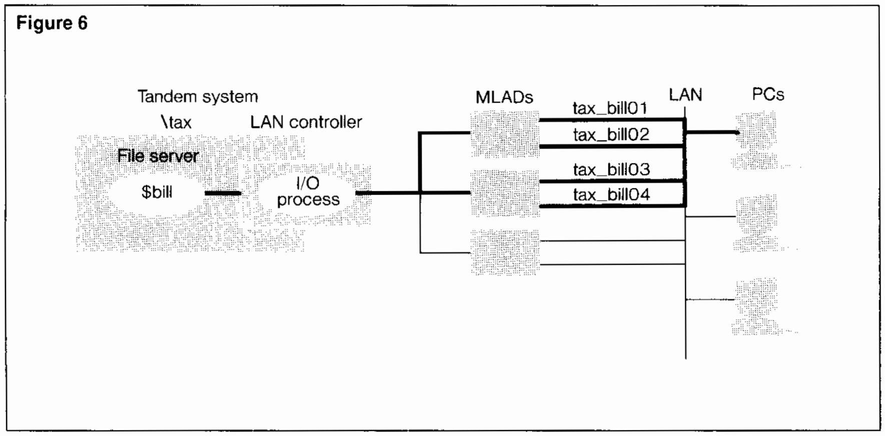


Figure 5.

In the simplest configuration, MLSRV provides access to PCs on a single LAN, using one LAN controller and one MLAD adapter.

Figure 6.

*Here MLSRV is configured to use four adapters and, thus, has four NETBIOS names. Assuming each adapter card has a limitation of 32 sessions and only one MLSRV process uses these adapters, 4 * 32, or 128, PC users can be connected in this configuration.*

Grouping all file-server files into subvolumes has important benefits for system management. System operators can move a file server's subvolume to a different disk, archive it daily, copy it, or analyze its disk space, all with standard Tandem tools. (There is no need for them to learn to use new tools, and indeed, the MULTILAN server provides no tools for these functions.)

The MULTILAN server gives PC users simple access to large, shared, mainframe printers. It diverts all spool output from PCs to the Tandem spooler and thence to Tandem printers. All Tandem spooler commands can be used on such jobs.

MLSRV has a user interface program, MLSRVCOM, that can be used to configure, manage, and control the operation of the various file-server processes. MLSRVCOM implements a superset of the server commands in the PC LAN Program, e.g., the SHARE command for sharing or "unsharing" directories, the FILES command for identifying open files or forcibly closing them, and the PAUSE and CONTINUE commands for temporarily suspending file-server activity. It also contains commands unique to the Tandem environment, e.g., commands to configure the file server, ask which PC users are currently connected, and get usage statistics.

Network Connectivity

MLSRV fully utilizes the features of the MULTILAN network, including fault tolerance, reliability, and high performance. It also uses many of the facilities of the GUARDIAN™ operating system and the EXPAND network.

MLSRV identifies itself on the LAN(s) by adding one or more NETBIOS names to the adapter card in the MULTILAN attachment device(s), or MLAD(s). This is controlled by configuration parameters set when MLSRV is run. At the time they connect to the file server, users specify one of those names.

In the simplest configuration, shown in Figure 5, MLSRV provides access to PCs on a single LAN, using one LAN controller and one MLAD adapter card.

In Figure 6, MLSRV is configured to use four adapter cards and, thus, has four NETBIOS names. Assuming each adapter card has a limitation of 32 sessions and only one MLSRV process uses these adapters, 4 * 32, or 128, PC users can be connected in this configuration.

MLSRV generates its NETBIOS names from the Tandem system (node) name, the Tandem process name, and a numeric suffix. In Figure 6, the file-server process named $bill running on Tandem node \tax has NETBIOS names tax_bill01, tax_bill02, tax_bill03, and tax_bill04. This algorithm allows users to know what the NETBIOS names for a file server are and guarantees the uniqueness of the NETBIOS names, no matter what the LAN configuration is.

Another possible configuration is shown in Figure 7. Here, three LAN controllers are attached to the same LAN. This configuration provides better performance and better availability in the event of a failure. Note that the NETBIOS names are unique on the LAN, even though multiple controllers are in use.

Again, multiple MLADs per controller are in use. MLSRV has added NETBIOS names that can be used by the PCs at connection time. For example, if a PC issued the command

NET USE N: \\tax_bill01\money,

it would use MLAD number 1 and controller $lam1. If another PC issued the command

NET USE N: \\tax_bill07\money,

it would use MLAD number 6 and controller $lam3.

MLSRV can communicate with multiple LANs, as shown in Figure 8. Since the LANs are distinct, the NETBIOS names on each LAN can be the same and the user PCs can access the file server without name conflicts.

MLSRV allows PC users on different LANs to share the same files under normal DOS rules. For example, the effects of a DOS lock on a file by a user on LAN $lam1 would be seen when any user on $lam2 tried to access that part of the file.

Figure 7

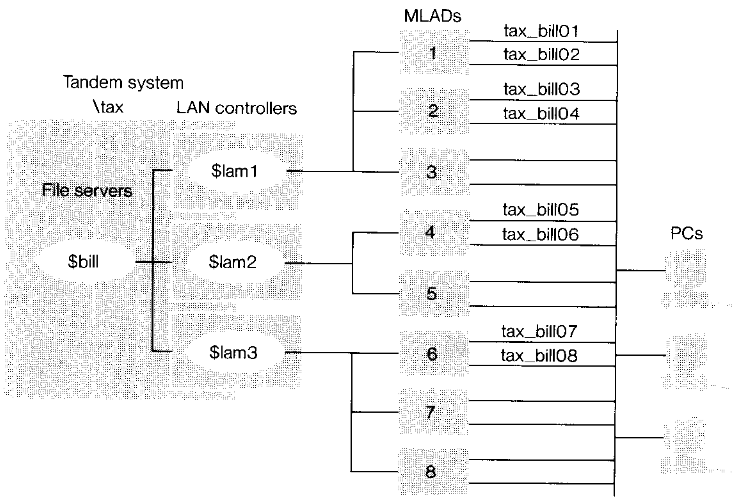


Figure 8

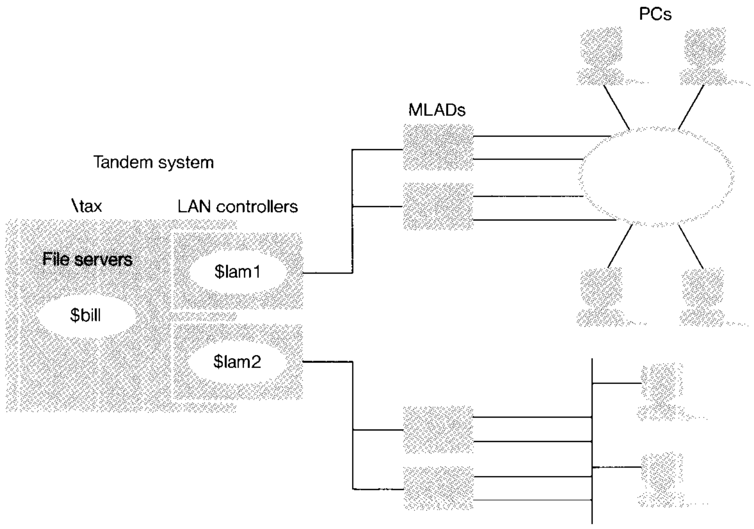


Figure 7.
Here three LAN controllers are attached to the same LAN. This configuration provides better performance and availability than the configuration in Figure 6.

Figure 8.
MLSRV can communicate with multiple LANs. Since the LANs are distinct, the NETBIOS names on each LAN can be the same and the user PCs can access the file server without name conflicts.

Figure 9

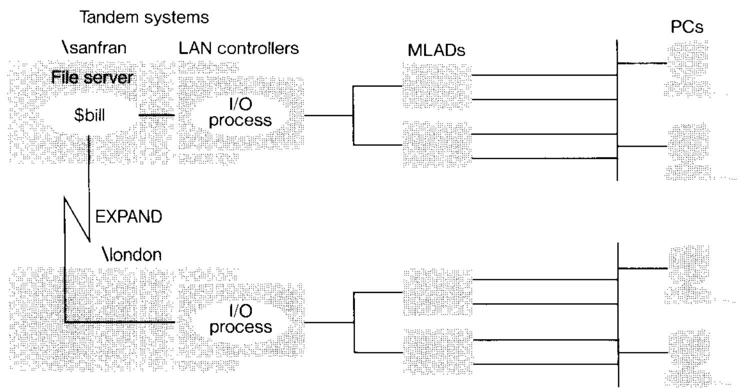


Figure 9.
LANs can be connected to any node in the Tandem EXPAND network. This configuration is most useful when the Tandem nodes are connected via high-speed lines (such as FOX), although PC users anywhere in the world can access the same DOS files controlled by one MLSRV process.

Finally, LANs can be connected to any node in the Tandem EXPAND network, as shown in Figure 9. This configuration is most useful when the Tandem nodes are connected via high-speed lines (such as FOX, Tandem's fiber-optic extension), as is often true when different departments each have their own node. Although the Tandem nodes are usually geographically close, they are not required to be. PC users in San Francisco and London could access the same DOS files controlled by one MLSRV process.

Fault Tolerance

The MULTILAN server and the Tandem architecture provide fault tolerance for many potential failures. In most instances, PC users do not notice that a failure has occurred. In some instances, they do notice a failure but are able to continue after making some manual recovery.

CPU Failure. CPU failure is handled by Tandem's NonStop architecture. Each file-server process is a process-pair. The primary process checkpoints any changed state to the backup process. There is usually only one checkpoint per SMB command. If the primary CPU fails, the backup takes over. It then creates another backup and continues processing the last request.

Suppose, however, that a CPU failure occurred just as the primary MLSRV was sending a message to a PC. It would not be possible to tell whether the message was completed successfully or not. If the old primary did not succeed in sending the message, the new primary would have to send it. If the old primary did succeed in sending the message, the new primary would duplicate it.

The same type of problem arises for a message coming in from the LAN to MLSRV when a failure occurs: did the message successfully arrive, and was its arrival checkpointed satisfactorily? Messages would have to be retransmitted if they were lost, yet not duplicated if they had arrived. Since MLSRV depends only on the standard DOS 3.x, redirector, and NETBIOS code in the user PC, the special recovery code for this situation cannot exist in the PC.

MULTILAN solves the problem by providing code in the MLAD and in MLSRV that recognizes duplicate messages and retransmits lost messages after a failure. Sequence numbers in the messages provide the basis for this recovery mechanism. The MLAD has the responsibility for remembering messages in transmittal between MLSRV and each PC. MLSRV resends the last messages after a failure. If the messages have duplicate numbers, the MLAD recognizes this and discards them; if the messages are not duplicates, the MLAD passes them on.

Similarly, for messages arriving for MLSRV, MLSRV tells the MLAD what sequence number it expects and the MLAD retransmits any message of that number. The cost of this scheme is primarily MLAD memory: it must remember two extra NETBIOS messages per MLSRV process plus one NETBIOS message per PC connected to each MLSRV. The result, though, is that no message between MLSRV and the MLAD is ever lost or duplicated, and CPU failure is completely transparent to PC users.

Disk Failure. Disk failure is handled automatically with mirrored disks. The disk process allows a single disk I/O from MLSRV to be sent to each of two disks. If a disk fails, the other mirror continues without MLSRV being aware of the problem. When REVIVE is used on-line to bring mirrored disks back in step after a failure, the application using the disk (e.g., MLSRV) is unaware of the recovery.

MLSRV does not use the Transaction Monitoring Facility (TMF™) to maintain its key-sequenced directory file, as the file has a single (primary) key and updates to it are straightforward. MLSRV does contain code to ensure that, even under system failure (e.g., long-term power failure) or double failure (e.g., the loss of the primary and backup CPUs for the disk process or both mirrored disks), its internal database is consistent. For example, the DOS operation CREATE FILE requires MLSRV to ask GUARDIAN to create the file and put its name into the key-sequenced directory file. MLSRV does the operations in that order so that the worst that can happen is that an extra (empty) file is created but its name is lost. DOS users would be unable to detect the inconsistency. (The opposite ordering could result in a DOS file appearing to exist but there being no equivalent GUARDIAN file.) Similar problems can occur when files are deleted or renamed.

Failure of the CPU Containing the Primary MULTILAN I/O Process. Another possibility is the failure of the CPU containing the primary MULTILAN I/O process. MULTILAN I/O processes are process-pairs, much like MLSRV processes. Each half of the pair has its own path into the dual-ported LAN controller. If the CPU containing the primary MULTILAN I/O process fails, MLSRV switches to the backup MULTILAN I/O process and retransmits all messages it thinks are outstanding. These messages go to the same controller and MLAD that the original messages went to. The MLAD message-sequencing scheme described above handles any duplicate messages occurring in this situation. The result, again, is that PC users are unaware that the failure occurred.

The failures discussed so far are all transparent to PC users. A number of failures do affect the user, however.

MLAD Failure. MLAD failure or the failure of a LAN adapter card in the MLAD cause all communication through that MLAD or that adapter card to be lost. If this happens, MLSRV terminates all sessions, releases any locks, and closes any open files that were opened through the failed MLAD. Then, MLSRV makes itself available through another MLAD by adding another NETBIOS name to it.

Typically, PC users recognize a failure when DOS displays an error message (such as "network device no longer exists"). The users' recovery cannot be automatic, but it is usually simple (assuming another MLAD is available or the failing MLAD had only a transient error and has recovered). Users disconnect from the server with

NET USE N: /d

and reconnect to it, for example, with

NET USE N: \\tax_bill03\money.

MULTILAN Controller Failure. MULTILAN controller failure occurs, for example, when the I/O process cannot communicate with the controller, when the controller cannot communicate with the MLAD, or when the controller hardware malfunctions.

In these situations, MLSRV is unable to use the controller or any of the MLADs on that controller. MLSRV performs recovery actions (such as closing files) for all MLADs attached to the controller and periodically tries to reestablish communication with the failed controller. (MLSRV reestablishes communication almost immediately if the failure is transient.)

As with MLAD failure, PC users using MLSRV through the failed controller are apprised of the failure by a DOS error message. Recovery here consists of using another controller, i.e., referring to MLSRV by one of its names accessible through another controller. For example, users could say

NET USE N: /d

and then

NET USE N: \\tax_bill07\money.

Power Failure. Power failure can affect a number of different components. In Tandem processors, battery backup provides power to the CPU memories for up to two hours. When power is restored, MLSRV resumes execution. Power is not maintained in the MULTILAN controller or MLAD; a failure of power to these units appears (to MLSRV) to be a transient failure and the recovery action described above occurs.

LAN Component Failure. Finally, the LAN components (e.g., PC adapter cards, transceiver or head end, and LAN cable) can fail. In most instances, no automatic recovery exists for this kind of failure. The components have to be repaired or replaced before the users affected by the failure can continue. When a LAN component fails, MLSRV cleans up its resource tables and periodically tries to communicate with the failed devices so that, when they return to service, the users can continue.

Expandability

MLSRV runs on processors of varying power and supports any size of Tandem disk. It supports multiple LANs, controllers, and MLADs. It supports large numbers of users, connections, sessions, open files, and spool files. More file-server processes can be created dynamically whenever required.

MLSRV's Integration with GUARDIAN

MLSRV's primary purpose is to be a pure DOS file server (i.e., to implement SMB protocol exactly). This allows PC users to work exclusively within the DOS environment.

For users who are also knowledgable about Tandem systems, MLSRV provides some degree of integration between the DOS and GUARDIAN environments. It allows DOS users direct access to GUARDIAN files and spooler jobs via DOS syntax and commands. Naturally, some restrictions apply, as SMB cannot be implemented to 100% accuracy here.

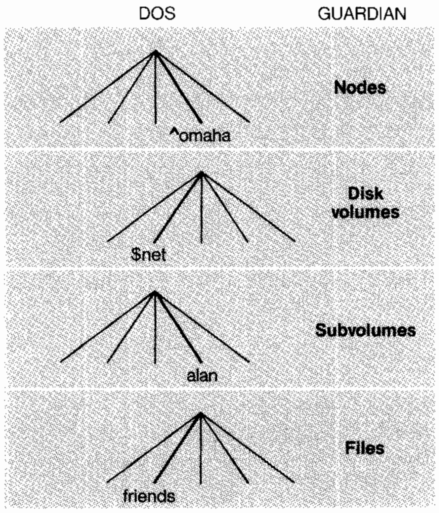
For security, PC users are required to be logged on to the Tandem system (using the MULTILAN RM/PC program). All MLSRV operations on behalf of any PC user are done under the user's GUARDIAN user ID.

Gaining Access to GUARDIAN Files

MLSRV allows GUARDIAN objects to be named and accessed. To facilitate this, the Tandem name space appears as a DOS directory structure of nodes, disk volumes, subvolumes, and files. At the file level, Tandem file codes appear as (optional) DOS file-name extensions.

To satisfy DOS rules for naming and reserved characters, the separators in GUARDIAN names must be changed to their DOS equivalents: the DOS "\" replaces the Tandem separator "." and the DOS "^" replaces the Tandem node character "\". (Refer to Figures 10 and 11.) DOS commands operate on Tandem objects as shown in Figure 11. (See the *MULTILAN User's Guide* for details.)

Figure 10



Tandem file
\omaha.$net.alan.friends

DOS file
^omaha\$net\alan\friends

Figure 10.
The DOS directory and subdirectory representation of GUARDIAN nodes, disk volumes, subvolumes, and files.

The GUARDIAN file type determines how a file can be used. Discussed fully in the *MULTILAN User's Guide*, the restrictions are summarized below.

Structured Files. Structured files can only be read sequentially.

Figure 11

(a) To list all Tandem nodes:

```
DIR \^*

Volume in drive G is GUARDIAN
Directory of G:\

^MIAMI      <DIR>    5-21-87    11:24a
^DETROIT    <DIR>    5-21-87    11:24a
^OMAHA      <DIR>    5-21-87    11:24a
^PHOENIX    <DIR>    5-21-87    11:24a
^CHICAGO    <DIR>    5-21-87    11:24a
^NEWYORK    <DIR>    5-21-87    11:24a
^LA         <DIR>    5-21-87    11:24a
```

(b) To list all GUARDIAN files starting with F on the node \OMAHA, volume $NET, and subvolume alan:

```
DIR \^OMAHA\$NET\alan\F*

Volume in drive G is GUARDIAN
Directory of G:\$NET\ALAN

FIONA     101    2526   5-13-87   10:47a
FOO       000       7   5-11-87    2:56p
FRIENDS   101    4566   3-31-87    1:34p
               3 File(s)        0 bytes free
```

(c) To list all edit files (code 101) starting with F:

```
DIR $NET\alan\F*.101

Volume in drive G is GUARDIAN
Directory of G:\$NET\ALAN

FIONA     101    2526   5-13-87   10:47a
FRIENDS   101    4566   3-31-87    1:34p
               2 File(s)        0 bytes free
```

Figure 11.

Listing a directory of GUARDIAN files with the DOS command DIR. (a) Listing all Tandem nodes. (b) Listing all files on a particular node, volume, and subvolume whose names start with F. (c) Listing all of those files that are edit files.

Edit Files. Edit files can be read or written only sequentially. DOS users can COPY, TYPE, or PRINT GUARDIAN edit files since these DOS commands operate sequentially. PC editors that satisfy the sequentiality restriction can be used to edit GUARDIAN edit files. (As not all PC editors can satisfy the restriction, fully verify that capability before using the program on a GUARDIAN file.)

Unstructured Files. Unstructured files have no restrictions. Sometimes using an unstructured file's GUARDIAN name when accessing it with MLSRV is more appropriate than using its DOS name, even though the two methods are almost equivalent.

For example, a GUARDIAN unstructured file can be shared between a PC program and a Tandem application program more easily than a file with a DOS name, since both ends can easily know its name. (The Tandem end opens $data.foo.file, while the PC program opens $data\foo\file.)

Another reason to use the file's GUARDIAN name is to allow use of facilities that MLSRV does not itself implement. (For example, MLSRV can access a partitioned unstructured GUARDIAN file that was created with FUP (File Utility Program), but it cannot create a DOS file with multiple partitions.)

Gaining Access to the Spooler

The main reason MLSRV allows PC users access to Tandem spooler files is to allow a spool job (created by some Tandem program) to be printed on a PC's local printer. (This is the opposite of MLSRV's more usual DOS print-server function, which directs PC spool files *to* the Tandem spooler.)

At the naming level, the spooler appears as a DOS directory tree. The root is a spooler supervisor; under it are its collectors; under the collectors are the locations; and under the locations are the jobs, which appear as DOS files. The spooler job status (e.g., RDY, OPN, or PRT) is turned into a three-character filename extension. How users access the spooler is shown in Figure 12.

Again, restrictions apply; the main one is that only sequential reading of spool jobs is possible.

Conclusion

It is essential that the data PC users share over LANs be saved securely and reliably and that it be continuously accessible across geographical and physical boundaries. Tandem's MULTILAN server, MLSRV, is a highly reliable, fault-tolerant, distributed DOS file server designed to meet these requirements.

References

IBM DOS 3.3 Technical Reference. International Business Machines Corporation.

IBM PC Local Area Network Program User's Guide. International Business Machines Corporation.

IBM PC Network Technical Reference. International Business Machines Corporation.

MULTILAN User's Guide. Tandem Computers Incorporated. Part no. 84058.

System Description Manual. Tandem Computers Incorporated. Part no. 84017.

Alan Rowe joined Tandem in 1980. Since then, he has developed operating system software, the MULTILAN server, and data communications software. Before joining Tandem, he spent six years developing operating system software for digital telephone switches and four years developing operating system software for a fault-tolerant multiprocessor.

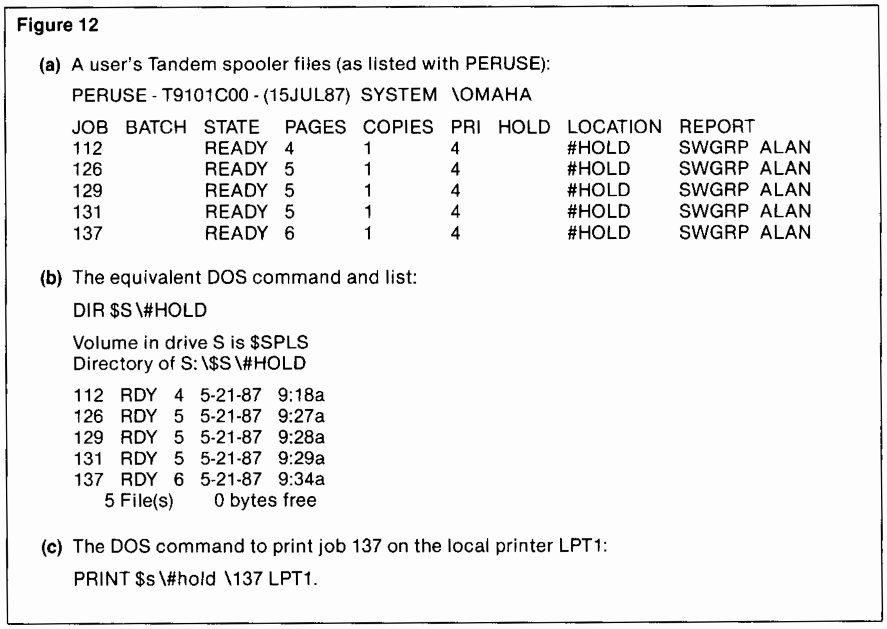


Figure 12

(a) A user's Tandem spooler files (as listed with PERUSE):

```
PERUSE - T9101C00 - (15JUL87) SYSTEM \OMAHA

JOB  BATCH  STATE  PAGES  COPIES  PRI  HOLD  LOCATION  REPORT
112         READY  4      1       4          #HOLD     SWGRP ALAN
126         READY  5      1       4          #HOLD     SWGRP ALAN
129         READY  5      1       4          #HOLD     SWGRP ALAN
131         READY  5      1       4          #HOLD     SWGRP ALAN
137         READY  6      1       4          #HOLD     SWGRP ALAN
```

(b) The equivalent DOS command and list:

```
DIR $S\#HOLD

Volume in drive S is $SPLS
Directory of S:\$S\#HOLD

112  RDY  4  5-21-87  9:18a
126  RDY  5  5-21-87  9:27a
129  RDY  5  5-21-87  9:28a
131  RDY  5  5-21-87  9:29a
137  RDY  6  5-21-87  9:34a
     5 File(s)    0 bytes free
```

(c) The DOS command to print job 137 on the local printer LPT1:

```
PRINT $s\#hold \137 LPT1.
```

Figure 12.

Gaining access to Tandem spooler files. (a) Listing them with the Tandem utility PERUSE. (b) Listing them with the DOS command DIR. (c) Sending them to the local printer with the DOS command PRINT.

Using the MULTILAN Application Interfaces

There are two MULTILAN application interfaces for communication between PCs and a Tandem host: the workstation 6530 emulation interface (WS6530) and the workstation process-to-process interface (WSPTP). WS6530 is suited for some applications and WSPTP for others. It is important to understand how the two interfaces work and how they are used before selecting one for programming.

This article describes the interfaces and discusses some of the application design and programming considerations that apply to each. Its examples are based on some of the designs employed within MULTILAN itself. To benefit from the discussion, readers should have a general knowledge of MULTILAN and its architecture. (See the accompanying article,"Introduction to MULTILAN.")

Overview

WS6530

The WS6530 interface allows Tandem application programs to treat a PC as a 6530 terminal in the usual process-terminal or master-slave relationship. Using it, the programs can perform operations such as OPENs, WRITEREADs, and CLOSEs just as they would to a 6530 terminal. The host side is implemented within the MULTILAN I/O process. On the PC side, the end recipient of the resulting 6530 data stream is usually PCT.EXE, the terminal emulator supplied with MULTILAN; however, application programmers can substitute their own PC programs to process the 6530 data stream.

Treating a PC as a 6530 terminal allows programmers to use any existing Tandem application program that writes to a terminal. The program can communicate with terminals or PCs. If programmers write their own programs using WS6530, they can use the same procedure calls from the GUARDIAN 90 operating system that they use for terminal I/O. They can also use the familiar programming style they use for all Tandem applications.

On the PC side, programs use the Communications Control System (CCS) library, supplied by Tandem, to interact with the Tandem program. CCS provides I/O routines (e.g., dc_open, dc_read, dc_write, and dc_close), modelled on GUARDIAN routines, that obviate the need for direct communication with the MULTILAN MS-DOS device drivers.

WSPTP

The WSPTP interface provides a general, powerful, peer-to-peer bidirectional message system between PCs and a Tandem host. Using it, any application running on a Tandem mainframe can communicate with any PC that uses NETBIOS (Network Basic Input Output System).

WSPTP provides the mechanism for NETBIOS sessions to be established between PC programs and Tandem applications, and for NETBIOS datagram and broadcast messages to be sent and received over the LAN. The same message system, NETBIOS, is used on both the Tandem and the PC sides. Generally, this means that application programmers can code and debug both sides together.

It is possible to share much of the source code between the PC and the Tandem implementations when a common language such as C or Pascal is used. There are some differences between the implementations, however. The NETBIOS "call," for example, is an interrupt on the PC and a call to WRITEREAD under GUARDIAN 90. Minor differences also occur in the format of the NETBIOS Control Block. These differences, discussed in detail later, can generally be hidden by a machine-specific covering routine and/or by conditional compilation.

NETBIOS can be used in any PC that has a LAN card and associated NETBIOS software. The PC LAN Program (which supports file servers such as the MULTILAN server, MLSRV) uses NETBIOS, as do the MULTILAN MS-DOS device drivers, LAN6530.SYS and RMPCDRVR.SYS. PC users can use NETBIOS independently, or they can share the use of the LAN with these programs. (Note that neither the PC LAN Program nor the MULTILAN PC drivers need be loaded to run a NETBIOS program on the PC.) Figure 1 illustrates the use of NETBIOS by the PC components of MULTILAN.

MULTILAN uses WSPTP internally, between the host and PC sides of its Resource Manager (MLRM and RMPCDRVR.SYS) and between MLSRV and the MS-DOS redirector.

Figure 1

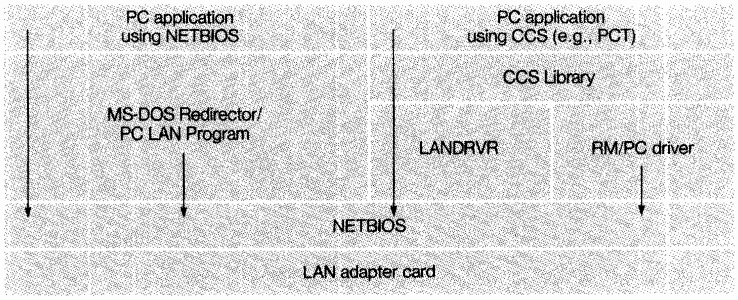

PC application using NETBIOS
PC application using CCS (e.g., PCT)
CCS Library
MS-DOS Redirector/ PC LAN Program
LANDRVR
RM/PC driver
NETBIOS
LAN adapter card

Figure 1.
How the PC components of MULTILAN use NETBIOS.

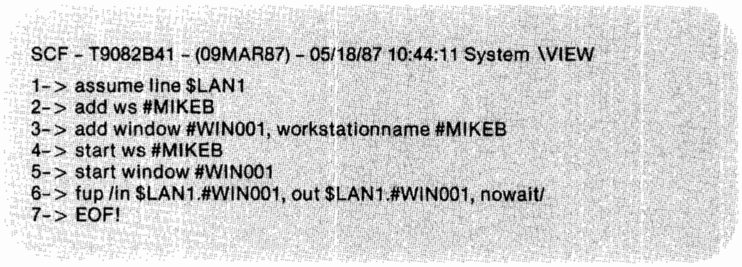
Using WS6530

The interface between a PC and WS6530 on the host is an MS-DOS device driver, LANDRVR. Tandem supplies the driver in the file LAN6530.SYS. To communicate with LANDRVR, PC programs must use the CCS Library, a set of Lattice C library routines used by PCT.EXE. This section discusses some aspects of using the CCS Library. Its examples use CCS to set up the Tandem host and the PC so they can communicate.

Using CCS to issue I/O calls to the MS-DOS LAN device drivers is straightforward. The *Communications Control System Programmer's Guide* describes these calls in detail.

Note that only a Lattice C version of the CCS Library is available with MULTILAN. If programmers use other languages, they must obey the Lattice C conventions for parameter passing and program linkage.

Figure 2



```
SCF - T9082B41 - (09MAR87) - 05/18/87 10:44:11 System \VIEW
1-> assume line $LAN1
2-> add ws #MIKEB
3-> add window #WIN001, workstationname #MIKEB
4-> start ws #MIKEB
5-> start window #WIN001
6-> fup /in $LAN1.#WIN001, out $LAN1.#WIN001, nowait/
7-> EOF!
```

Figure 3

```
#include "dc.h"       /* CCS Library header file */
dc_file datacom;      /* Attribute array used to communicate */
                      /* with LANDRVR */
int error;

main ()
{
    /* Initialize fields in attribute array */
  dc_finit(datacom);

    /* Set window name */
  dc_set(datacom,dc_wname,"\\VIEW.$LAN1.#WIN001");

    /* Open driver and make sure the open worked */
  if ((error = dc_open(datacom,"LANDRVR")) | |(error = dc_wait(datacom, - 1))) {
    printf("dc_open( ): error %d\n",error);
    exit(- 1);
  }
    /* Communication loop */
  for(;;) {
  .
  .
  .
    /* Calls to dc_read( ), dc_write( ), and dc_wait( ) */

  .
  .
  .
  }
    /* Close driver and make sure the close worked */
  if ((error = dc_close (datacom)) | |(error = dc_wait(datacom, - 1))) {
    printf("dc_close( ): error %d\n",error);
    exit(- 1);
  }
  exit(0);
}
```

Figure 2.

Using a Subsystem Control Facility (SCF) session to configure a static window on a Tandem system and start FUP.

The LAN I/O process is $LAN1, the PC (or workstation) is MIKEB, and the window is #WIN001.

Figure 3.

One example of a PC program (written in Lattice C) that opens a static window and uses it.

Windows

MULTILAN allows multiple WS6530 streams, called *windows*, per PC. If a PC windowing package (such as Microsoft Windows or Quarterdeck's DESQview) runs several terminal emulation windows, each window corresponds to a WS6530 data stream. In other words, terminal emulation windows on the PC correspond to WS6530 windows. (WS6530 windows are analogous to subdevices in other Tandem communication access methods.)

WS6530 windows can be static or dynamic. *Static* windows are required whenever a Tandem application must know the device name in advance, such as when terminals are configured for the PATHWAY transaction processing system. Tandem's Subsystem Control Facility (SCF) is used to create them. Refer to the *Subsystem Control Facility Reference Manual* for details.

Dynamic windows are created by MULTILAN software when requested by the PC (without requiring special preconfiguration on the Tandem side), and they are automatically deleted by MULTILAN when they are no longer needed. They are usually created with the PCT's resource_name option. MULTILAN supports up to four dynamic windows per workstation.

An Example of a Static Window. A static window might be used to enable a PC program to communicate with Tandem's File Utility Program (FUP) over a LAN. To do this, a programmer would first configure the window on the Tandem system and start FUP. The SCF session in Figure 2 shows how this would be done for the PC MIKEB, the static window #WIN001, and a LAN controlled by the LAN I/O process $LAN1. Note that with MULTILAN, as with other Tandem access methods, the names of the objects configured are upshifted automatically.

Next, on the PC, the programmer would run the program that reads and writes to the window. The PC program would use the CCS Library to communicate with LANDRVR. The program fragment in Figure 3 contains one version of the code, written in Lattice C.[1] (Note that the CCS Library requires a window name to have the special format "\\VIEW.$LAN1.#WIN001" where \VIEW is the EXPAND node name, $LAN1 is the name of the line, and #WIN001 is the window name that was configured in SCF. This string must be in uppercase.)

[1] Refer to the *Lattice C Compiler Programmer's Reference Manual* for syntax conventions.

An Example of a Dynamic Window. If the programmer used a dynamic window to communicate with FUP, there would be no need to use SCF on the Tandem system and no need to start FUP manually. To use the window, the programmer would first log on at the PC using the PC program RMPCCOM, which is the user interface to the Resource Manager for the PC (RM/PC). The MS-DOS command sequence in Figure 4 accomplishes this. In the example, the name MLAN.USER represents a valid GUARDIAN 90 user who has permission to execute FUP.

Figure 5 illustrates the next step: a PC program, written in C, requests the MULTILAN software to create a dynamic window on the host system \VIEW and start FUP in that window. In this example, the PC code causes the MULTILAN software to create a process on the Tandem system and send it a startup message. The dynamic window is automatically created, and its file name is generated on the Tandem host and inserted into the startup message, for use by the host program.

The 6530 data stream from FUP looks the same to the PC regardless of whether the PC program uses static or dynamic windows. In the example just discussed, the program should expect to see the 4-byte sequence

<SOH>C<ETX>@,

which is the control sequence for placing a 6530 terminal in conversational mode. If it followed the 6530 protocol, the program would respond with an <ACK> after receiving this control sequence.

Sending Binary Data

Programs can use WS6530 to send binary data over MULTILAN. Programmers must ascertain that no control sequences are added during the binary transmission, however. For example, when a Tandem program issues SETMODEs, the PC may see them as control, or <SOH>C<ETX>@, sequences. If a program issues SETMODEs when it initializes (as is common), it should send a specific message to cause the PC to go into binary mode after the SETMODEs are done and before sending the binary data. The PC should not expect the binary data until it has seen this message. Users should be particularly careful when the Tandem program has a run-time library, as C does, since the run-time library can issue unexpected SETMODEs.

Figure 4

```
C:\>rmpccom
TANDEM MULTILAN RESOURCE MANAGER COMMUNICATION UTILITY . . .
RMDRVR1 opened.
?set machine name = mypc
Machine name set to “MYPC”.
?logon @ \view MLAN.USER,MLAN
Logged on @ \view; MLRM version B41.
?exit
RMDRVR1 closed.
```

Figure 5

```
#include “dc.h”       /* CCS Library header file */
dc_file datacom;      /* Attribute array used to communicate*/
                      /* with LANDRVR */
int error;
main ()
{
    /* Initialize fields in attribute array */
  dc_finit(datacom);
    /* Set Tandem host node name */
  dc_set(datacom,dc_hname,“\\VIEW”);
    /* Set name of program MULTILAN is to start */
  dc_set(datacom,dc_rname,“FUP”);
    /* Set startup parameter string */
  dc_set(datacom,dc_sps,“/PRI 150, CPU 4/”);
    /* Open driver and make sure the open worked */
  if ((error = dc_open(datacom,“LANDRVR”)) | | (error = dc_wait(datacom, – 1))) {
    printf(“dc_open( ): error %d\n”,error);
    exit( – 1);
  }
    /* Communication loop */
  for(;;) {
  .
  .
  .
    /* Calls to dc_read( ), dc_write( ), and dc_wait( ) */
  .
  .
  .
  }
    /* Close driver and make sure the close worked */
  if ((error = dc_close (datacom)) | | (error = dc_wait(datacom, – 1))) {
    printf(“dc_close( ): error %d\n”,error);
    exit( – 1);
  }
  exit(0);
}
```

Figure 4.
The MS-DOS command sequence used to log on at a PC with the PC program RMPCCOM (the user interface to the Resource Manager for the PC, RM/PC).

Figure 5.
One way for a PC program (written in Lattice C) to request the MULTILAN software to create a dynamic window on a host system (\VIEW) and start FUP in that window.

If the PC program properly follows 6530 protocol, it has the advantage of being able to function correctly over an asynchronous line with minor modifications only. In fact, the only difference between the PC program fragment in Figure 3 and one that would use an asynchronous line is that the PC program would open ASYNC1 instead of LANDRVR.

Using WSPTP

This section explains how to use WSPTP, the NETBIOS equivalent in MULTILAN, to communicate between Tandem host applications and PCs. WSPTP is described in detail in the *MULTILAN Programmer's Guide.* NETBIOS for PCs is described in IBM's *PC Network Technical Reference.*

NETBIOS Basics

NETBIOS provides two main types of communication: sessions and datagrams. It also provides informational routines.

Sessions. In the NETBIOS protocol, a session is a logical connection between two named endpoints on a LAN, for example, between two PCs or between a PC and a Tandem host application. Both ends can send or receive messages simultaneously, and both sides use the same set of message primitives. (This differs from the GUARDIAN 90 requester-server model in which requesters call WRITEREAD and servers call READUPDATE and REPLY.)

The underlying protocols of NETBIOS ensure that messages sent over sessions are delivered reliably between the two LAN adapter cards or that the sender is told if a message failed to get through.

Sessions are used in all current MULTILAN software. Since programmers using WSPTP commonly use sessions, the sections following this one concentrate almost exclusively on them.

Datagrams. Datagrams are one-directional messages that go from one station to another, to a group of other stations, or for broadcast datagrams, from one to all. Datagrams have no associated protocols to guarantee successful sending and receipt of messages. Any application that uses datagrams and depends on reliable message delivery must provide its own protocol. When reliable delivery is required or one end needs to detect the loss of the other end, sessions are recommended.

For example, a stock exchange application could use datagrams appropriately to broadcast the current Dow Jones average to all stockbroker PCs. It would not matter if a message occasionally failed to get through. The application would use sessions to record the trades initiated by the PCs, however.

Informational Routines. NETBIOS and WSPTP provide some routines to ask about the state of various aspects of the network (e.g., ADAPTERSTATUS and SESSIONSTATUS). Refer to the appropriate manuals for further details about these.

Names and Sessions

In NETBIOS, PCs and Tandem host applications must each have a NETBIOS *name* that is 16 characters long and unique on the network. A process chooses a name and puts it on the network by using the NETBIOS ADDNAME command. If it succeeds (that is, if the name it has chosen is unique and there is room for it), NETBIOS returns the name_number corresponding to the name. This name_number is used in subsequent NETBIOS CALLs.

As discussed above, a session is a logical connection or, more specifically, a connection between two NETBIOS names. Before a session can be established, both ends must have added their names. The session is established after (1) one end has issued a NETBIOS LISTEN to indicate that it is prepared to accept an incoming CALL, (2) the other end has issued the CALL, and (3) the CALL has been matched with the LISTEN.

An MLSRV process adds its name as it starts up and issues a LISTEN immediately after the ADDNAME completes. A PC using MLSRV adds its name after it is booted and the PC LAN Program is run. It issues its CALL as required to establish a session. Once a session has been created, NETBIOS and WSPTP assign a unique session_number, to be used in subsequent messages.

If a listener is to communicate with multiple callers as MLSRV does, it must post multiple LISTENs. The LISTEN must be posted before the CALL is issued, or the CALL fails. Contention between callers can arise if more than one wants to call a particular listener at the same time. NETBIOS matches the first CALL with the LISTEN and builds a session; if another CALL occurs before another LISTEN is issued, that CALL fails.

Programmers can use one of two easy techniques to avoid caller contention. One technique is to have the caller delay after the CALL has failed and retry a few times. This technique is used between RM/PC and MLRM.

The second technique is to have the listener issue multiple outstanding LISTENs. Then, while one session is being established, other LISTENs are available for further CALLs. The number of LISTENs required depends on the time each one takes. For MLSRV, two LISTENs have proved sufficient.

Once a session has been established, programs can SEND or RECEIVE on it. The NETBIOS interface buffers SENDs and RECEIVEs so that either the SEND or RECEIVE can occur first. Generally, once a session has been set up with a matching LISTEN/CALL pair, that session is used for many SEND/RECEIVE messages. For example, once a session has been set up between the MS-DOS redirector and MLSRV (with the NET USE command), all file I/O between MS-DOS and MLSRV uses SENDs and RECEIVEs over that session.

A program can establish several sessions under one name and then issue the NETBIOS RECEIVEANY command to accept data sent to it on any of the sessions under that name. MLSRV uses this feature to get the next available message from any user PC. It establishes one session per PC and issues a RECEIVEANY for each of its names to get the next available message.

A session is removed by a HANGUP from either end. Rebooting a PC or powering it off causes a HANGUP on all of its sessions. The other end is notified of the loss of the session when it receives a special error code on its next NETBIOS operation having that session number. A name is removed with DELETENAME.

Figure 6

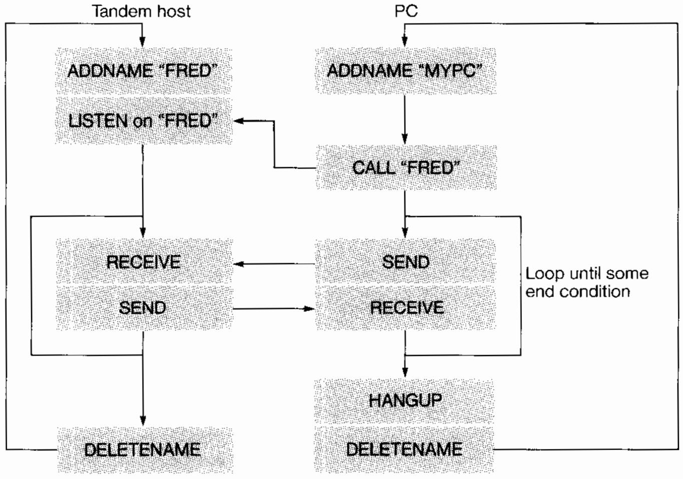


Figure 6 illustrates a typical NETBIOS sequence.

A Tandem process "owns" the names and sessions it creates, and as no other process can use them, they are secure. When a process dies, its names and sessions are deleted.

On the PC side, any PC program can use any NETBIOS name or session. An exiting PC program should remove its names and sessions explicitly or a reboot will probably be necessary.

A NETBIOS timeout on a RECEIVE or RECEIVEANY is harmless (much like a GUARDIAN timeout on a "nowait" I/O operation). A SEND timeout ends the session, however, so users should construct their applications so that SENDs never time out. SEND and RECEIVE timeouts are specified with the CALL or LISTEN command for the associated session. On the PC side, specify "no timeout" for SENDs and RECEIVEs. On the Tandem side, MULTILAN software interprets a zero timeout on a SEND as the maximum NETBIOS timeout (about 2 minutes).

Figure 6.

A typical NETBIOS sequence.

Requester-Server Model

Although sessions are bidirectional, in many instances the PC is a requester and the Tandem host is a server. In this model, the following sequence occurs:

1. The Tandem application issues an ADDNAME as it initializes itself.
2. The Tandem application LISTENs.
3. The PC issues an ADDNAME when the application is run (often immediately after rebooting, if the program is called from an AUTOEXEC). This name is necessary to identify the PC on the LAN but is never CALLed (and the PC never LISTENs).
4. The PC CALLs the Tandem application. It can generally assume that the Tandem end has already issued the ADDNAME, since the Tandem application is always running. It can often assume that the Tandem end has LISTENed also.
5. The PC SENDs a message to the Tandem application, which issues a RECEIVE or RECEIVEANY to get it.
6. The Tandem application SENDs the reply, and the PC issues a RECEIVE to get it.

Using NETBIOS Via WSPTP

A PC calls the various NETBIOS routines by passing a data structure called a Network Control Block (NCB). The NCB contains items such as the command code, error response, and number of the session or name. It also contains a pointer to the address of the data to be sent or the address into which the data is to be read. When it has set up the NCB, the PC program issues the machine instruction INT 5C to "call" the NETBIOS code.

When using WSPTP, programmers must use a slightly different method. First, open WSPTP by opening the file [\node.]$lan.#WSPTP, where \node.$lan is the name of an I/O process connected to the LAN on which the PCs reside. The open can specify that future I/Os be wait or nowait (the open itself must be waited). Syncdepth should be 0 or 1: syncdepth = 1 implies that the File System is to retry GUARDIAN I/Os if the CPU in which the primary I/O process resides fails.

Then, issue NETBIOS requests with a WRITEREAD to the resultant file number. Place the NCB at the beginning of the buffer in the WRITEREAD. Include any user data to be sent or received immediately after the NCB. If the I/O is nowait, call AWAITIO, in the usual way, to complete the I/O.

Close the I/O process when it is no longer needed. The last CLOSE from a process deletes any remaining names and sessions created with that I/O process.

NETBIOS and WSPTP both use NCBs, but the forms of the NCBs are slightly different. In WSPTP, some NCB fields are unused and often must be zero, and the width of some fields differs from that of the fields in NETBIOS. For example, in NETBIOS, only 256 sessions can be used. In WSPTP, 256 would be unacceptably small, so the 1-byte session_number field is augmented with a 2-byte session_number extension field. As a consequence, whenever session_number is required or returned, Tandem applications should move both parts.

A coding note: It is often convenient for the Tandem program to pass session_number as a single item rather than as two separate fields (e.g., when passing it as a parameter or storing it in various data structures). Use a TAL INT(32) for this. Form the INT(32) from the two fields, by using $DBLL, after LISTEN returns them. Retrieve the two fields with $HIGH and $INT, and write them back to the NCB (e.g., before issuing a SEND command).

The same two-field technique is used for name_numbers, and the same rules apply.

The file NCBDECS is provided as part of MULTILAN. It contains TAL™ (Tandem Application Language) definitions for the NCB and for useful literals, such as command codes and error numbers. Programmers using languages other than TAL should base their programs on this file, using the same names wherever possible.

The TAL program fragment in Figure 7 uses WSPTP to perform a NETBIOS ADDNAME operation.[2]

Design Considerations

Nowait I/O. Since WSPTP operations are File System calls, they can be nowait. Nowait is specified by the OPEN flags, and not by setting the nowait bit in the NCB as it would be for a PC using NETBIOS.

If more than 15 nowait operations are to be performed, WSPTP must be opened multiple times. There is no association between a particular open of WSPTP and any NETBIOS entity. For example, a RECEIVE on a specific session can be issued on any of the process's WSPTP files that are currently open; it does not have to be issued on the file number used when the session was created.

As usual for nowait I/O, consider the amount of system or process resource (e.g., LCBs and PFS) to be consumed.

On the PC side, nowait NETBIOS operations are also possible. If the PC uses a windowing or multitasking package, be sure that use of nowait I/O is compatible with it. Specifically, do not use the POST address for nowait I/O in a PC in such situations.

Message Protocols. Even though NETBIOS sessions are reliable, session-based messages between a PC and the Tandem application can still be lost or duplicated. For example, if a CPU fails, GUARDIAN 90 may reissue the I/O in a backup CPU, thus duplicating the message. Often, applications need to address the possibility of lost or duplicate messages.

In the Tandem environment, any application that uses NETBIOS has a PC end and a Tandem end, both coded as part of the same application. For such applications, use an end-to-end protocol built on top of NETBIOS sessions, and include syncIDs, loss detection, duplicate detection if duplicates matter, and retries for possible errors.

Figure 7

```
?NOLIST, SOURCE NCBDECS, list

BEGIN
  STRUCT .ncb(NCB^template); !NCB to send to WSPTP
  LITERAL ncb^length = $LEN(ncb);
  INT .wsptp^name[0:11];        !Name of LAN I/O process
  INT wsptp^num;                !File number of open #WSPTP
  INT(32) name^number;          !Number of added name

    wsptp^name ':=' "$LAM1    #WSPTP      ";
    call OPEN(wsptp^name, wsptp^num);
    if <> then . . .
      -- Handle open failure

    !Many fields in the NCB have to be zero, so zero the whole structure
    ncb.NCB^command := 0; !The first field in the NCB
    ncb.NCB^command[1] ':=' ncb.ncb^command[0] FOR ncb^length - 1; !bytes

    !Set up NCB for an ADDNAME
    ncb.NCB^command  := NCB^Cmd^Add^Name;
    ncb.NCB^name    ':=' "VIEW_BILL01    ";
    ncb.NCB^version  := Current^NCB^Version;

    !Issue ADDNAME
    call WRITEREAD(wsptp^num, ncb, ncb^length, ncb^length);

    !Did the WRITEREAD work?
    if <> then . . .
      -- Call COMMERROR to handle error on LAN

    !Did the ADDNAME work?
    if ncb.NCB^retcode <> Good^Return then
      -- Handle NETBIOS error

    !Get resultant name_number from the two places in the NCB
     name^number := $DBLL(ncb.NCB^num,
                          ncb.NCB^num^ext);
END;
```

Figure 7.
A TAL program fragment that uses WSPTP to perform a NETBIOS ADDNAME operation.

Note that MLSRV is an exception to this rule. The PC end is not written by Tandem, so no end-to-end protocol is possible. Instead, messages are sent reliably between the Tandem host and the MULTILAN Attachment Device (MLAD), and losses and duplicates are detected by cooperating code in the MLAD software and MLSRV.

Do *not* use the MLAD "slot sequence number" scheme used by MLSRV. It provides fewer benefits, is more complex, and costs more in MLAD resources than PC code written to perform the equivalent function.

[2] Refer to Tandem's *Transaction Application Language (TAL) Reference Manual* for syntax conventions.

Error Handling. Writing a resilient NETBIOS program requires careful thought. Both NETBIOS and the Tandem host can return errors of various types (e.g., hard and retriable).

On the Tandem side, error handling is a two-level problem. Both Tandem file errors and NETBIOS errors can occur. Tandem errors are typically returned by the I/O process or controller; NETBIOS errors are returned by the MLAD or controller.

Tandem errors are returned, in the usual way, from WRITEREAD or AWAITIO. Call COMMERROR, a part of MULTILAN, for suggested recovery actions appropriate for the File System error (e.g., CPU failure, power failure, I/O process switch, controller error, or MLAD failure). COMMERROR returns the status, saying, for example, whether the error is fatal and whether it should be retried (and, if so, whether to retry immediately or after a delay). In some instances, recovery from hard errors is possible; for example, if an MLAD fails, it is possible to use ADDNAME to add the names of the failed MLAD to another MLAD and then continue. With a properly designed end-to-end protocol, the application is unaware of many of these problems.

If COMMERROR indicates that no Tandem error has occurred, examine the NETBIOS error code in the NCB. The NETBIOS manual suggests the appropriate recovery action, such as retry.

Use a state-machine to handle all WSPTP I/O. This approach is often the only one suitable for handling errors by retrying I/Os.

On the PC side, only NETBIOS errors can occur. For these, follow the manual's recommended recovery actions. No procedure equivalent to COMMERROR is available; handle errors individually, according to the NETBIOS command issued.

Names and Sessions. The total number of names and sessions each MLAD can support depends on the type of LAN to which it is attached. Since these are finite resources, use them with care.

The two resources are linked. For example, if a Tandem application were to communicate with 33 PCs simultaneously, it would need 33 sessions. For a LAN limited to 16 names and 32 sessions, two MLADs would be required and, hence, two names (assuming the MLAD were not used by other programs). If the MLAD were already in use, 33 sessions might require more MLADs and more names.

There are a number of ways to reduce the number of names and sessions required. One would be to multiplex use of the sessions. That is, have only one session between a PC application and the Tandem end and multiplex all the information on the session. (Of course, this often requires a more difficult protocol and might result in poor code structure.)

A second solution would be to keep sessions short. For example, the RM/PC program builds a session to the MULTILAN resource manager (MLRM), sends a message, and breaks the session. It uses only one name and one session per LAN, no matter how many PCs are in use, and it retries when another RM/PC program or PC is using the session. This technique is appropriate when messages are infrequent and the conversation is very short.

A third solution would be to develop automatic algorithms. For example, if the quantity of names or sessions required cannot be predicted, write the Tandem application so that it can adjust to accommodate a varying quantity. For example, consider a Tandem application having a name TMON and a set of names TANDEM1, TANDEM2, ... Normally, the PC would be coded to CALL (and build a session to) one of the TANDEMx names. Develop a protocol in which the PC tries to CALL to each name in turn, and if it fails because there are no sessions available, have it CALL and then SEND to TMON asking the Tandem process to add another TANDEMx name. This works because, once a LISTEN is posted by TMON, a CALL is guaranteed to succeed.

Choosing Names. An application requires a 16-character NETBIOS name on either side before it can successfully use sessions. Note that any 8-bit character is valid (including control characters and null) and lowercase is interpreted differently from uppercase. For names that are not naturally 16 characters, space filling on the right is a common convention. It is also useful to make every character printable (and readable), especially if the name is to be entered by users, displayed, or found in a debugger.

Names must be unique. How can they be chosen? On the PC side, the PC is given a NETBIOS name by the action of NET START or, equivalently, by RMPCCOM SET MACHINE NAME. In both instances, the PC user decides on the machine name and the ADDNAME guarantees its uniqueness.

It is usually very dangerous to use one of the existing PC names, since one cannot predict how that name is used by the PC application that added it.

For applications running in the PC, the easiest way to choose another NETBIOS name is to derive it from the PC machine name. PC machine names, like all NETBIOS names, are 16 characters long; however, when the PC is attached via MULTILAN, the PC name is effectively restricted to a maximum of 7 ASCII printable characters plus 9 trailing spaces, padding the name to 16 characters. (This is because MULTILAN forms a Tandem subdevice name from the PC name.)

Names are usually generated by taking the 16-character PC machine name and substituting a different last character. This is the technique used within NET START to differentiate the various PC LAN Program components and within RM/PC to identify the LAN driver. For example, if an application chooses "X" as its last character, a PC named by the RMPCCOM command "SET MACHINE NAME = MYPC" would form the NETBIOS name

```
"MYPC           X".
```

To avoid conflict with existing PC code, do not choose any of these as the last character: null, space, %03, %05, "L", "d", or "T".

On the Tandem side, unique name selection is less obvious. Generally, the PC is a requester and the Tandem end is a server, so the PC has to know the Tandem name (but not vice versa). The simplest solution is to pick a name and hard code it in both the Tandem and PC applications. This works for a single instance of the application only.

Another solution is to derive the NETBIOS name from some equivalent Tandem name, such as that of the Tandem process. For example, a Tandem process called $FRED might add the NETBIOS name

"FRED   ".

Since $FRED is unique,

"FRED   "

is likely to be unique too (although this is not guaranteed). Since a LAN can often span multiple Tandem nodes, the name is more likely to be unique if the Tandem node name is appended in some way. Thus \CUP.$FRED might add the name

"CUP_FRED   "

and \SUN.$FRED would add

"SUN_FRED   ".

Some derivative of this technique is used by most of the MULTILAN software.

When multiple related names are needed (e.g., when an application needs names in more than one MLAD), the simplest approach is to append digits to the generated name. MLSRV adopts this technique, adding names such as

"VIEW_FRED01   "

and

"VIEW_FRED02   ".

Note that, with techniques such as these, the PC end cannot know the NETBIOS name in advance. It has to find it out by some other technique, such as getting it from the user via some parameter string or asking another process with a well-known name which name it should use.

Performance

The performance of an application is usually highly dependent on the size of the messages transmitted between the PC and Tandem host. It is usually preferable to send one larger message rather than send many small messages. The overhead per message of small messages results in poorer performance.

MULTILAN allows buffers of nearly 32 Kbytes. Generally, for highest throughput, use large buffers; however, bigger is not necessarily better. Large buffers occupy often scarce PC memory, scarce Tandem resources (e.g., PFS), and resources inside the MULTILAN components (e.g., MLAD and controller memory). Model the application first, allow the buffer sizes to be varied easily, measure the application, and tune the application by varying the buffer size.

Use other appropriate techniques for improving performance also. For example, double buffering (issuing two RECEIVEs or RECEIVEANYs) allows one request to be processed while another is arriving.

Topology

WSPTP applications can exploit MULTILAN's alternatives for a flexible topology. If large numbers of PCs are being serviced and the MLAD session limit is reached, multiple MLADs are required. To handle the MLADs, use one name per MLAD and issue commands such as ADDNAME and LISTEN on each.

Also, if PCs on different LANs are to be serviced or if, for better performance or fault tolerance, multiple controllers exist on the same LAN, multiple I/O processes must be opened. Note that Tandem's EXPAND network facilitates communication with LANs on different Tandem nodes.

Conclusion

MULTILAN provides a powerful framework to allow PCs and Tandem host applications to cooperate. The design and programming considerations discussed in this article should help application designers to choose the appropriate MULTILAN application interface and design their applications appropriately.

References

Communications Control System (CCS) Programmer's Guide. Tandem Computers Incorporated. Part no. 37271.

EM6530PC Terminal Emulator Programmer's Guide. Tandem Computers Incorporated. Part no. 37168.

Lattice C Compiler Programmer's Reference Manual. Vols. 1 and 2. Lattice, Incorporated.

MULTILAN Programer's Guide. Tandem Computers Incorporated. Part no. 82488.

MULTILAN User's Guide. Tandem Computers Incorporated. Part no. 84058.

PC Network Technical Reference. International Business Machines Corporation.

Subsystem Control Facility (SCF) Reference Manual. Tandem Computers Incorporated. Part no. 84012.

Tandem Application Language (TAL) Reference Manual. Tandem Computers Incorporated. Part no. 82581.

Mike Berg joined Tandem's Large Systems Marketing Support Group in July 1986. He has over ten years of application and system design experience. Mike has a Ph.D. in Mathematics from Dartmouth College.

Alan Rowe coauthored this article with Mike Berg and wrote the accompanying article, "An Overview of the MULTILAN Server."

Sizing the Spooler Collector Data File

The Tandem spooler subsystem is a set of processes and procedures that serve as a buffer between an application writing to a print device and the device itself. This article discusses how to size the spooler collector data file (the file in which data to be output to a print device is collected). It is intended primarily for system managers responsible for configuring the spooler subsystem.

Basic Spooler Principles and Terminology

The following is a brief description of how the Tandem spooler subsystem works. The discussion is limited to high-level spooling, i.e., the level of spooling in which commands such as

```
>COBOL /IN SOURCE, OUT $S/
```

are used (where $S is the name of a spooler collector). For an in-depth discussion of the spooler, refer to the references listed at the end of the article.

When a process wishes to spool data to a print device, it writes the data to a *collector process*, which stores it in a disk *data file.* Several collectors can be active at any time, and usually (but not necessarily) one is named $S. Each collector manages its own data file.

The collectors are controlled by a *supervisor process*, usually named $SPLS. Several supervisors, each controlling its own spooling system, can be active at any time. Figure 1 shows the relationships of the processes within the spooler subsystem.

When a process first writes to a collector, a new spooler *job* is registered in the supervisor. (If the user process opens a collector and closes it again without writing any data to it, no new job is defined.) A job consists of all the data written to the collector from the time the collector is opened until it is closed. One collector can handle up to 128 jobs simultaneously. One supervisor can handle a maximum of 4096 jobs, but can be configured to handle a lower number.

When a collector is closed, the job is normally ready to be printed. (This is subject to, among other things, the availability of a print device.) The supervisor schedules the job for printing according to the job's priority. This priority can be altered at any time with the SPOOLCOM or PERUSE utilities.

When the job is scheduled to be printed, a *print process* (supplied by Tandem or written by the user) reads the data file by calling the procedure PRINTREAD, which returns one print line at a time. This operation is blocked: a new block is read when all the lines from the previous block are returned.

The print process then writes the data to the designated print device. When all requested copies of the job are printed, the print process informs the supervisor that the job is printed. If the job is not set up to be held in the system after printing (set HOLDAFTER on), it is deleted, and the space it occupied in the data file becomes available for other jobs.

Collector Unit Size

The collector writes spooled data to the disk data file in blocks, each of which is called a *unit*. Each job requires a minimum of one unit but can, of course, require many more, depending on job/unit size. When a collector needs more disk space for a job, it allocates one unit at a time from the data file.

Unit 0, the first unit in the file, is used internally by the collector, so it is not available for data storage. For this reason, a collector can show a data file utilization above 0%, even if no jobs are present. (Note that collector unit allocation, performed by the collector, is logical, while disk extent allocation, performed by the disk process, is physical.)

A collector is configured into the spooler subsystem with the SPOOLCOM utility program. The command that defines the unit size of the data file used by a collector is

```
COLLECT <collector>, UNIT <number>
```

where <number>, multiplied by 1024, is the unit size (in bytes). The minimum value of <number> is 2; the maximum, 32,767. If no UNIT parameter is specified when the collector is configured, the default value is 4 (equal to a unit size of 4096 bytes, or two pages).

One collector can handle a maximum of 8192 units. It is important to keep this in mind when determining the extent size of the data file. (This point is discussed in more detail in the section, "Data File Sizing and Allocation.")

Figure 1

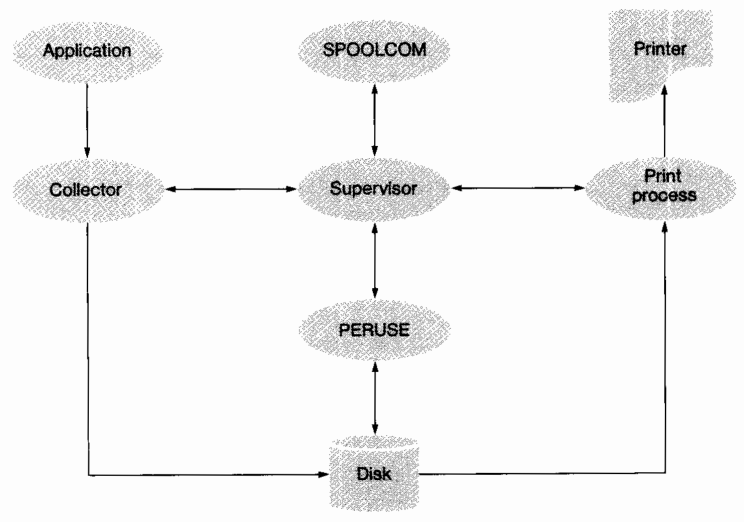


Figure 1.
Spooler components.

To utilize data file space efficiently, decide the value of <number> for the UNIT parameter carefully. If the value is large and the majority of spooled jobs is fairly small, the last (or only) allocated unit of each job may contain much unused space. On the other hand, if the majority of the jobs is large, a small unit size causes more frequent unit allocation, which increases the service time for the job. The unit size also determines the maximum usable size of the data file.

If the size of the spooled jobs varies greatly, to utilize disk space efficiently, configure several collectors with different UNIT values (e.g., $SS, $SM, and $SL, for "small," "medium," and "large," with UNIT values of 2, 4, and 8). If users adhere to a certain discipline (i.e., not using $SS for large jobs or $SL for small jobs), this is one way to manage the available disk space effectively.

Data File Sizing and Allocation

When the spooler is first started ("cold" start), the collector writes a "dummy record" to the calculated EOF position. This causes the disk process to physically allocate all the disk file extents for the data file at one time. As the collector later writes spooled data to the data file, it allocates space for the jobs as needed, one unit at a time, using a bit map to keep track of free and allocated units. Units for the same job do not have to be contiguous: the first unit of a job contains the relative byte address (RBA) of the job's next unit, and so on.

The data file *cannot* be enlarged after the collector has been started. The only alternative is to create a larger file, possibly change the unit size of the collector, and start the spooler cold.

When determining the physical size of the data file, remember that a collector can handle, at most, 8192 units. If the size is exactly

(unitsize, in bytes) * 8192,

the collector can make use of all the available space. If the size is larger, the space beyond unit 8191 is never used.

If the size of the data file is smaller, the collector allocates only as many units as fit in the available space, which decreases the number of jobs as well as the size of the jobs the collector can handle. For example, if (1) the supervisor is configured to handle 3000 jobs, (2) one collector is used, and (3) the size of the data file permits the allocation of only 2048 units, the maximum number of jobs the spooler is able to handle is 2047 (assuming each job requires only one unit of space, otherwise the number is even less). Table 1 shows the maximum usable data file sizes for UNIT values 2 through 14.

Table 1.
Maximum data file sizes for UNIT values 2 through 14.

UNIT value	Maximum data file size (Mbytes)	Number of disk pages
2	16.77	8,192
3	25.16	12,288
4	33.55	16,384
5	41.94	20,480
6	50.33	24,576
7	58.72	28,672
8	67.10	32,768
9	75.49	36,864
10	83.88	40,960
11	92.27	45,056
12	100.66	49,152
13	109.05	53,248
14	117.44	57,344

The exaggerated examples in Table 2 may help to illustrate what happens if the data file size does not "balance" the UNIT value chosen. In a data file allocated with EXT (2,2) and a UNIT value of 10, the collector is able to store, at most, five jobs since there is room for only six units in the file when all its disk extents are allocated. (The collector reserves one unit for internal bookkeeping.)

In a data file allocated with EXT (1000, 1000) and a UNIT value of 2, the collector is able to use all 8192 units, but note that the EOF pointer is set at 16,777,216, or roughly half the possible file allocation. (If the file were preallocated to ensure that the required disk space were available, the "unused space" could, of course, be reclaimed with a FUP DEALLOCATE command, directed to the data file after once starting the spooler cold.)

When a job needs another unit and no more units are available in the data file, the collector generates a pseudo Error 45 (file is full). Note also that the collector does not make use of the DP2 file label attribute MAXEXTENTS.

Considerations for Data File Allocation

It makes sense to calculate the maximum file size needed for the data file and then allocate it in one extent. This way the disk remains less fragmented and there is no need for the disk process to switch between file extents.

Refer to Table 1 for the relationship between the UNIT value and file size. When creating the data file, use the appropriate value in the "Number of disk pages" column as the value of the primary extent. For example, for a UNIT value of 4,

>FUP CREATE <file>, EXT (16384, 0).

The value chosen for the secondary extent size does not matter when this method is used, since a secondary extent will never be physically allocated.

Note that performance is not affected if the size chosen does not permit all 8192 units to be used. This affects only the capacity of the collector (i.e., the size and number of jobs it is able to handle).

Spooler data files are typically files that "last forever" (or at least for the life of the system). It may therefore be a good idea to allocate the spooler data files from empty disks, immediately after labeling the disk pack. This further minimizes fragmentation of available disk space. (Make sure that the data files are not purged and reallocated every time the spooler is started cold, as this is unnecessary and could cause the file to be allocated in another area of the disk.)

Conclusion

When configuring the spooler system, follow these steps to utilize available disk space efficiently and ensure optimum spooler performance:

- Use UNIT values reflecting the size of the majority of spool jobs.
- If job sizes vary greatly, consider configuring several collectors.
- If space allows, size the data file EOF to be equal to (unit size, in bytes) * 8192.
- If disk space fragmentation is a concern, allocate the collector data file in a single extent.
- If the size of the data file is unchanged, do not purge and reallocate it when the spooler is started cold.

Table 2.
Values for two spooler collector data files allocated with a UNIT value of 10 and extent size specifications of EXT (2,2) and EXT (1000,1000).

Variable or entity	Calculation	Resulting value
EXT (2,2)		
PRIMARYEXT	2 (pages) * 2048	4,096
SECONDARYEXT	2 (pages) * 2048	4,096
MAXFILESIZE	PRIMARYEXT + 15 * SECONDARYEXT	65,536
UNITSIZE	10 * 1024	10,240
MAXUNITS	MAXFILESIZE / UNITSIZE	6
EOF (RBA)[†]	UNITSIZE * MAXUNITS	61,440
Unused space	MAXFILESIZE - EOF	4,096
EXT (1000,1000)		
PRIMARYEXT	1000 (pages) * 2048	2,048,000
SECONDARYEXT	1000 (pages) * 2048	2,048,000
MAXFILESIZE	PRIMARYEXT + 15 * SECONDARYEXT	32,768,000
UNITSIZE	2 * 1024	2,048
MAXUNITS	MAXFILESIZE / UNITSIZE	16,000
EOF (RBA)[†]	UNITSIZE * 8192	16,777,216
Unused space	MAXFILESIZE - EOF	15,990,784

[†] In the calculation of the "logical EOF," or relative byte address of the first unused byte of the data file, UNITSIZE is multiplied by MAXUNITS for the file allocated with EXT (2,2) and by 8192 for the file allocated with EXT (1000,1000). Since the collector can handle at most 8192 units, UNITSIZE is multiplied by the smaller value of (MAXUNITS, 8192).

References

Spooler Programmer's Guide. Tandem Computers Incorporated. Part no. 82394.

GUARDIAN 90 Operating System Utilities Reference Manual. Tandem Computers Incorporated. Part no. 84011.

System Procedure Calls Reference Manual. Tandem Computers Incorporated. Part no. 84047.

Acknowledgments

The author would like to thank David Cooper, Joe Massucco, Bob Sawyer, Scott Sitler, Tuomo Stauffer, and Steve Watanabe for their comments and encouragement.

Henry Norman joined Tandem in March 1982 as a support analyst in Stockholm, Sweden. Since then, he has supported the GUARDIAN operating system and subsystems in Tandem's Western Systems Support Group and has worked on DP2 in Software Development. Before joining Tandem, he spent ten years in systems programming for transportation systems, including two years as a consultant specializing in performance measurement and tuning of large mainframe installations.

C00 TMDS Performance

The Tandem Maintenance and Diagnostic System (TMDS) is a hardware diagnostic system providing an easy and consistent interface for system managers and customer engineers to maintain, diagnose, and upgrade hardware subsystems. TMDS analyzes and diagnoses system problems as they occur, often identifying failures before they affect system performance.

This article provides a performance overview of the C00 software release of TMDS and its component parts. This includes performance analysis of event processing, analyzing subsystem faults, running diagnostic routines, and the effect that TMDS has during a cold load of a system. For more information on TMDS's features and uses, refer to the last issue of the *Tandem Systems Review* (Blain, et al., 1987).

Since TMDS is an integral part of all Tandem NonStop systems, it must operate in an efficient and optimal manner. The performance studies in this article show that TMDS has more than met those criteria, offering needed fault analysis and diagnostics with little impact on system cost. In fact, the background activity for all TMDS processes to analyze an event takes up as little as 0.05% of CPU usage and costs as little as 34 milliseconds (ms). This efficient usage of system resources ensures that TMDS will be a successful addition to the Tandem product line.

TMDS Structure

TMDS has four NonStop processes: $ZLOG, $ZELM, $ZENO, and $ZMOM. $ZLOG receives all TMDS events generated on a system and, in turn, logs those events to a key-sequenced log file named ESLOG with an alternate-key file ESLOG0. ESLOG and ESLOG0 reside on $SYSTEM.SYSTEM.

The process $ZELM monitors the log file ESLOG and performs memory reallocation on the log to ensure that there is always space for incoming events.

The process $ZENO is the monitor of the TMDS system. Through the TMDS end-user interface, a user can interact with $ZENO to extract information from the TMDS system or to run diagnostics on hardware subsystems through application monitors. $ZENO also interacts with the other TMDS processes to control TMDS actions.

$ZMOM is the fault-analysis process. It identifies events in ESLOG that represent possible hardware fault conditions and, through $ZENO, initiates the appropriate fault analyzers (FAs) to examine the problem. These FAs may correct the problem through their own action or may have to notify operators of the condition. This notification takes the form of a message to the console or a phone call through the Remote Maintenance Interface modem to Tandem's On-line Support Center. Figure 1 shows the component parts of TMDS and how they interact with each other.

Figure 1

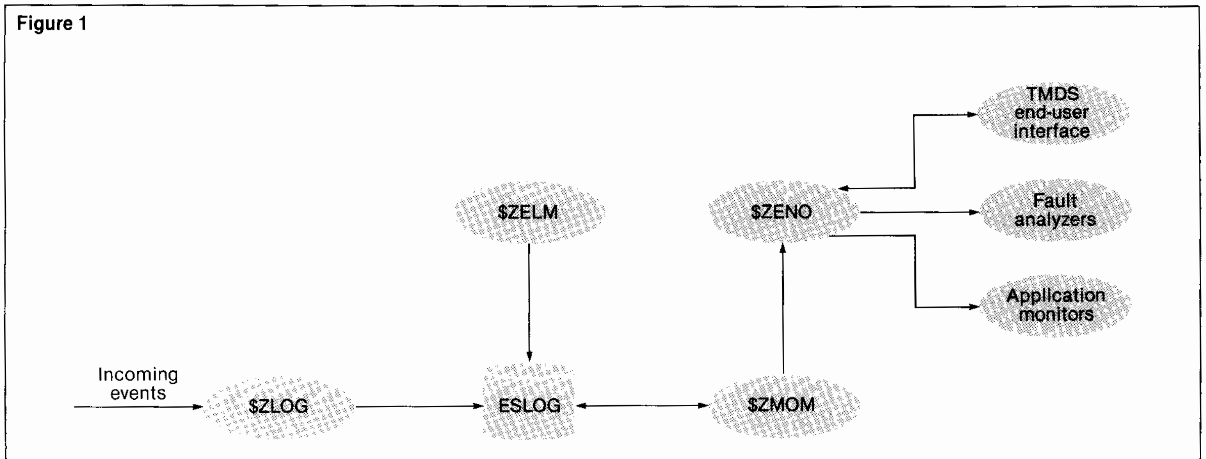


Figure 1.
Component parts of TMDS.

Measuring the Cost of Event Throughput

A hardware subsystem informs TMDS of a hardware error or change in the state of the hardware by generating an *event*. An event is a message in a standardized form that can be read by the $ZLOG process and logged to the TMDS event log ESLOG.

The cost for TMDS to log and examine an event is an important metric for determining the overall cost of running TMDS on a system. For the following performance tests, events were generated on a *stand-alone* system where the exact cost per process could be identified (a four-processor NonStop VLX system isolated from network traffic was used). A stand-alone system is a system dedicated to the use of the processes under measurement, so that measurements may be done without the interference of extraneous system uses.

Each event measured was generated by means of a specialized event generator. The event generator produced an event consisting of the normal event structure, but containing a comment for the body of the event. This event generator varied the rate and size of events sent to TMDS, thereby allowing examination of all possible event conditions to be found on the system. The use of an event generator also allowed the examination of the overhead of the TMDS system without looking at the overhead associated with the actual generation of real events from any subsystem.

To determine the TMDS cost of each event generated on the system, several tests were undertaken. First, a measurement of a quiesced system was taken for a period of 10 minutes. This measurement gave a background reading of the cost of TMDS processes when there is no activity on the system. Second, measurements were taken with events generated on the system over the same time frame as the quiesced system measurement. These events were generated with different sizes and rates. From these measurements it was possible to determine the exact TMDS cost for an event generated on the system.

Figure 2

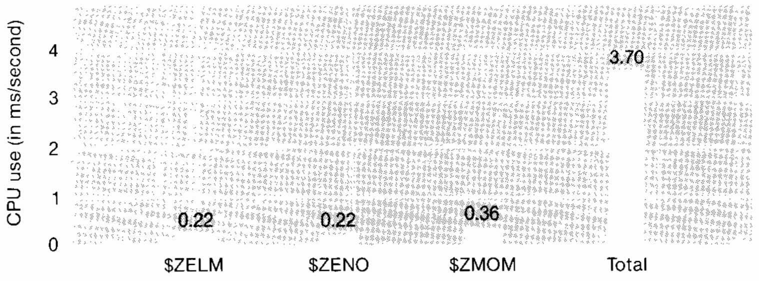


Figure 2.

Quiesced VLX CPU utilization. The total CPU consumption for all primary TMDS processes was 0.80 ms of CPU busy time per second. The "total CPU" bar indicates total CPU usage by all processes in a single quiesced processor.

Quiesced Systems

A quiesced system is a system on which there is no activity other than that which goes on in the background of the processors. Although the operating system and other system processes such as TMDS continue to check their system resources on the quiesced system, no explicit process activity is taking place.

TMDS activity on a quiesced system consists of the TMDS processes checking ESLOG for any event activity. $ZELM and $ZMOM continuously check the log file for any events that may have been generated on the system. These events are examined by $ZMOM to identify potential problems, while $ZELM monitors the log and deletes any old or redundant events. $ZENO is activated when a problem is identified on the system and FAs are called into play or when the TMDS end-user interface is opened to examine the condition of the system. On the quiesced system, $ZENO shows some activity as it reads system messages from its $RECEIVE file. Since no activity is taking place on the system, $ZLOG will remain inactive as its sole purpose is to log events as they occur to the event log.

Figure 2 shows the CPU utilization for the primary TMDS processes on a quiesced system. The total CPU consumption for all primary TMDS processes was 0.80 ms of CPU busy time per second on the VLX system. The backup processes for TMDS use only 0.16 ms of CPU time per second. This means that all TMDS activity on the measured VLX system accounted for 0.96 ms, or less than 0.1% of the CPU busy time of a single processor.

Process Measurement

For all of the TMDS processes, the event rate and size had minimal impact on the cost of processing each event. As long as the event rate was in the range of one event or less per second, the CPU cost per event remained stable. Systems operating under the C00 software release produce events in the range of 1 to 20 events per hour. This average throughput rate is highly dependent on the system size and configuration and will increase as more subsystems produce greater numbers of events. Even so, an active system with 16 CPUs, 31 disks, and 2 tape drives fell easily within this throughput range, averaging only ten events per hour. The one event per second throughput represents a rate that will seldom occur and is easily handled by the TMDS system. As shown later in this article, TMDS can easily process more than twice that number before experiencing any performance degradation.

Figure 3 shows the CPU cost per process involved in processing a TMDS event on a VLX system. All the TMDS primary processes account for 29.5 ms of CPU time with the $ZLOG backup process accounting for another 4.5 ms. For a system with ten events per hour, the cost amounts to only 340 ms of CPU time spread out over an hour.

When discussing "CPU cost per event" generated on a system, what is meant is the extra cost above the quiescent system cost per process. In other words, for the tests referred to in this article, all costs were computed by running a large number of events through the system, measuring the total CPU cost, subtracting the quiescent CPU cost for that measured period of time, and dividing the remainder by the number of events generated. The accuracy of measurement is increased by running a large number of events through the system for any single measurement.

When a TMDS event is generated on a system, it is read by the $ZLOG process and logged to the log file ESLOG. $ZELM reads the new event and, if ESLOG is 85% full, deletes any old or duplicate events in the log. $ZMOM examines the event to determine if any further action needs to be carried out on the event. If further action is necessary, $ZMOM informs the $ZENO process to call an FA into action. The event is then analyzed and the problem corrected or the appropriate action taken.

In this test environment, the events generated do not represent real problems. Therefore the $ZENO process is not activated when the events are examined. On active systems, one expects to see some $ZENO activity because of fault analysis and the running of diagnostics through the TMDS end-user interface. On VLX systems in development environments with approximately 100 users, $ZENO was measured in the range of 0.6 to 0.8 ms CPU busy time per second of CPU time.

High Throughput

One of the limiting factors in the throughput of the TMDS system is the rate of I/O to the event log. The event log file ESLOG is a buffered key-sequenced file with an alternate-key file containing three alternate keys. This file structure was designed to provide quick uniform access to any information in the event log. Since the ESLOG is generally measured in megabytes, this is a reasonable approach for the design of the file.

It is important that ESLOG not lose information on events generated on the system, as these events may be the only record of problems that have occurred on the system. Since ESLOG is a buffered file, there is a small risk of losing an event if the disk cache is, for some reason, not written to disk before a double CPU failure. This is a rare occurrence and shouldn't affect most systems, but may affect systems when multiple CPUs are frozen simultaneously for a system dump.

Figure 3

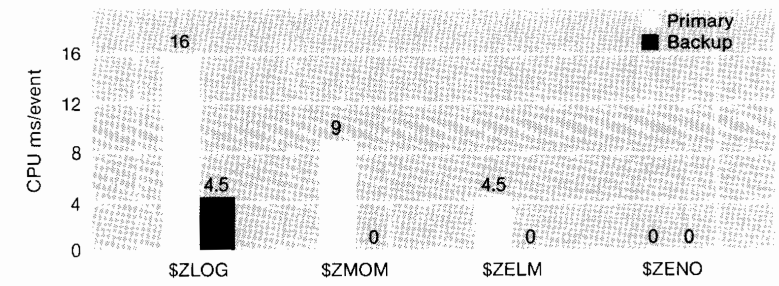


Figure 3.
CPU cost per process for processing a TMDS event on a VLX system. The primary processes account for 29.5 ms of CPU time with the $ZLOG backup process accounting for an additional 4.5 ms. The cost to a system with ten events per hour would amount to only 340 ms of CPU time spread out over an hour.

If this is a problem, the ESLOG file can be reconfigured dynamically or at SYSGEN time to unbuffer the file. Of course, this will adversely affect TMDS performance by forcing nine physical writes to disk for every event logged through the $ZLOG process. This limitation also affects the other TMDS processes that are reading and writing to the same log file and adversely affects all system performance.

Even with these limitations, TMDS, with a buffered ESLOG, measured an event throughput of 2.5 events per second on the VLX system. This event rate is high enough to handle most systems in a catastrophic environment and any system in normal situations.

Figure 4

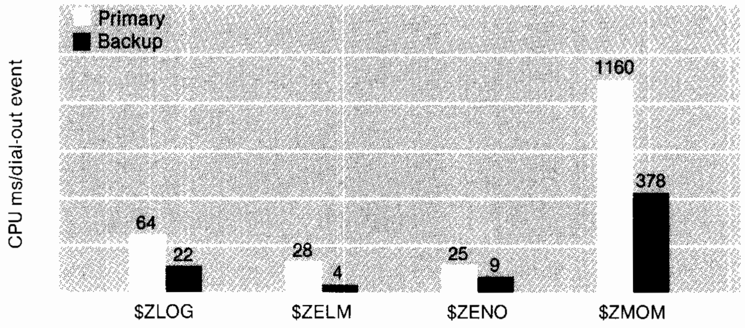


Figure 4.

CPU use while analyzing faults. Though $ZMOM uses substantial CPU time, the number of events needing analysis is usually quite low.

Fault Analysis

$ZMOM carries out fault analysis on all events entered into the TMDS logs. Depending on the type of event logged, $ZMOM performs an action on the event. This action can range from doing nothing at all, to trying to fix a problem, to dialing out for customer engineering assistance.

To get a general sense of what fault analysis on the TMDS system costs, it was decided to measure an event that would cause the creation of an FA to investigate a problem. The event chosen was an attempt to dial out to a remote system.

The dial-out event caused $ZMOM, with the help of $ZENO, to call the dial-out FA. The dial-out FA attempted the actual call, but in this case, the dial-out subsystem was disconnected so no actual call was placed. Since the attempt to call out was unsuccessful, the dial-out event case was marked "solved" or completed.

So, for each dial-out event, $ZMOM reads the event, opens a dial-out event case, and calls the dial-out FA. The dial-out FA attempts to begin the dial-out mechanism, realizes it is inoperative, and "solves" the dial-out event case. $ZMOM then moves on to the next event.

Figure 4 shows the CPU use for TMDS while analyzing faults. $ZMOM uses substantial CPU time. This is to be expected in a fault analysis environment. Although fault analysis is costly to run, the number of events needing analysis is usually quite low. More than a couple of events a day in need of such extensive analysis is rare. The actual CPU use by $ZMOM for any particular event is dependent on the health of the system and the severity of the problems the system is experiencing.

Memory Consumption

When memory use is measured on stand-alone test systems, a process's memory consumption represents the maximum amount of memory that can be used by that process. This is because there are no other processes resident to contend for the memory resources available. This is not a true indicator of the memory use by processes in an on-line system.

To better portray the actual memory use of the TMDS processes, several systems other than the stand-alone test systems were measured. Figure 5 shows the memory use on several systems operating under the C00 software release. The first two columns for each process represent the maximum swappable memory use of the process as measured on the stand-alone test system. The next two columns display the average number of swappable memory pages in use on active development systems that were measured. Of course these active systems are uncontrolled environments, so there is no way to understand the event load handled by TMDS on those systems.

Figure 5

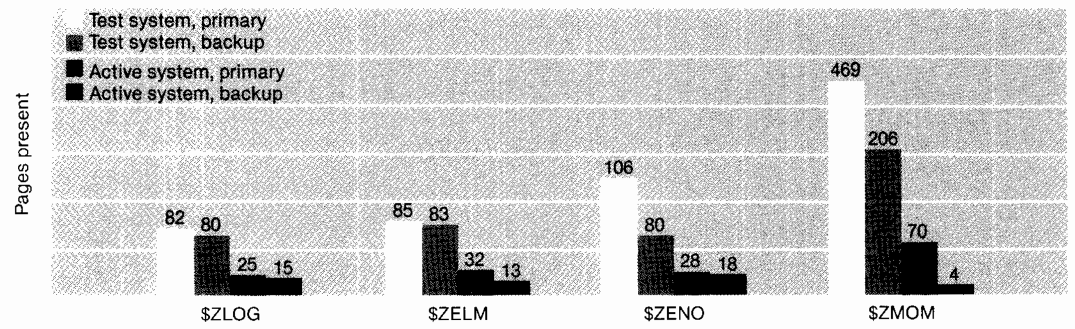


Figure 5.
Memory consumption. $ZLOG and $ZELM were close in their use of memory for primary and backup processes. However, $ZENO's and $ZMOM's memory use was much more varied, depending on the faults being analyzed and the diagnostics performed.

Given that uncertainty, all the active development systems were remarkably close in their use of memory for the TMDS processes. $ZLOG and $ZELM were consistent in their use of memory for their primary and backup processes. $ZENO and $ZMOM memory use was much more varied, depending on the faults being analyzed on the system and the diagnostics performed. $ZENO ranged between 28 and 60 pages of memory allocated to its primary process. Likewise, $ZMOM ranged between 70 to 600 pages of memory.

Since TMDS does not lock down any memory, all memory in use indicates the fault-analysis processes' diagnosing problems. The amount of memory in use is a real indicator of the work being done by TMDS.

Cold-Load Time

TMDS is started automatically at cold-load time through the TMDSAUTO process. TMDSAUTO starts all TMDS processes and initiates the logging of messages to the TMDS subsystem. TMDS minimizes the effect that events have on the cold-load time by restricting the start-up activities of $ZELM and $ZMOM. These restrictions minimize the effect that TMDS has on system startup time. All TMDS processes contribute less than 1 minute to startup time.

TMDS's impact on cold-load performance is caused primarily by the TMDS processes reading the log file ESLOG. This is done at cold-load time so event information will be available for analysis. Therefore, the delay in cold-load time is directly related to the size of the ESLOG and the number of events in the file. ESLOG typically can be a 2-Mbyte file containing approximately 4000 events. A file of this size allows for logging of one event a minute for 2.5 days before the file becomes full. The time needed to read a key-sequenced file of that size is approximately 30 seconds. Therefore, TMDS's total effect on cold-load time is be about 30 to 40 seconds.

To measure the actual impact of TMDS, a VLX system was cold loaded once with TMDSAUTO active and again without the TMDSAUTO process active. When TMDSAUTO was absent at system start-up time, TMDS processes were not started.

Figure 6

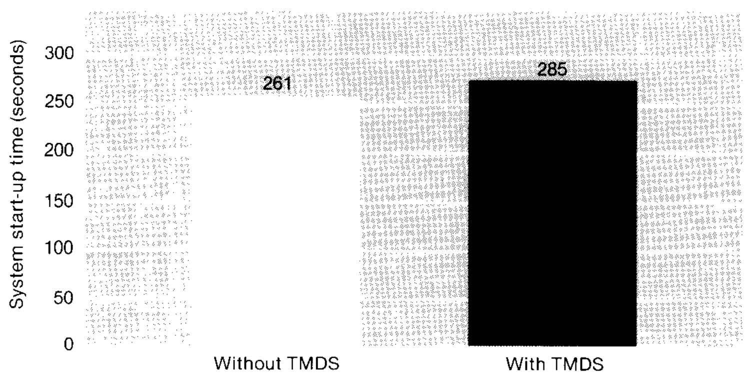


Figure 6.
Cold-load measurement results.

Each system had similar software and hardware configurations, which included the following:

- Four processors.
- Nine terminals with TACL™ (Tandem Advanced Command Language) prompts.
- Four comm lines.
- A spooler and other system processes.
- A PATHWAY system consisting of 30 terminals, two requesters, and one server.

The ESLOG was 1.8 Mbyte and was 66% full.

The cold-load time was measured from the time the processor cold-load key was pressed on the OSP until the PATHWAY system had completed its cold start. By the time the PATHWAY system was brought up on the system, TMDS had completed its start-up phase. Measuring the difference between cold loading the system with and without TMDSAUTO can give a rough measurement of the impact TMDS has on the system start-up time. Figure 6 portrays the cold-load measurement results.

TMDS was responsible for 24 seconds of cold-load time or a degradation of approximately 8% in cold-load time on this VLX system. These measurements indicate that the effect of TMDS on the cold load of a system is minimal compared with the benefits gained.

Diagnostics

To complete the performance analysis of TMDS, the impact of running diagnostic tests on a system under use was needed. The disk subsystem was chosen for these measurements because of the wide range of disk diagnostic tests available, and because the typical heavy use of disks makes any disk diagnostic test immediately apparent to the user of the system.

The target disk for the test was a V8 (4110) with 122 Mbytes of usable space. The disk was a mirrored DP2 disk configured for parallel writes on a VLX system. The disk was 88% full. To simulate a heavily used system, the mirror of the disk was taken down and then revived. This creates a situation where continuous reads are posted on the primary disk, thereby setting up a worst-case scenario for the diagnostic to be performed. Although this is not normally done on a production system, it meets the criteria of the test by creating a heavily used environment for purposes of comparison under load.

First, the revive was done on the disk without any diagnostic activity. This was measured with wall clock time from the point the revive was started until the disk activity halted. The revive was optimized to complete as quickly as possible.

Second, the revive was done with a diagnostic running concurrently. The diagnostic was the TMDS Disk Diagnostic Quick Test, looping forever. The Quick Test runs a variety of tests on the disk for as long as the revive is being carried out. The tests are all nondestructive tests, representing the type of tests run on a social system while the system is in use. Figure 7 shows the results of the revive measurements.

The initial revive (without diagnostics) was carried out in 13 minutes and 15 seconds. The revive with diagnostics was run in 27 minutes and 50 seconds. This is a difference of 14 minutes and 35 seconds, or a 110% increase in the time required to revive the disk.

This test is not indicative of all real-life situations, but still allows conclusions to be drawn. Diagnostics run on a system in use may have a severe impact on the performance of that system. A user of the system or disk under diagnosis may experience a noticeable degradation in response time while trying to access the disk under test. Different diagnostic tests have varying degrees of impact on the system. The Diagnostic Quick Test normally lasts for about 1 minute, but there are diagnostics that run for 30 minutes or longer. Any diagnostic that may cause serious performance degradation on a system should be run when the degradation will be least noticeable. In other words, even though many diagnostics can be run on a system that is currently under use, it is best to delay the tests until the system is at its lowest use to reduce the impact to users.

Figure 7

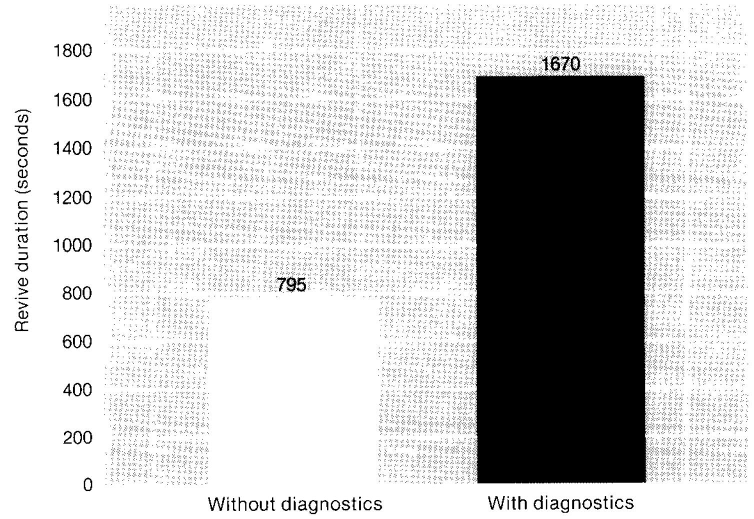


Figure 7.
Revive measurement results.

Conclusion

This performance study shows that TMDS handles the logging of events and the analysis of faults in an efficient and optimal manner.

Even though event throughput is typically seen in the range of 10 to 20 events per hour, TMDS event throughput is handled efficiently with event rates as high as one to two events per second. At these high throughput rates, all TMDS processes can log and analyze an event with as little as 34 ms of CPU time. For the average rate of ten events per hour, this means there will be only 0.1% CPU usage of the processor in which TMDS resides.

Fault analysis on the TMDS system provides a valuable tool for the evaluation of system events. Although the cost of providing such analysis may be over 1 second of CPU time, the information provided from this analysis is invaluable and will save the system operator and customer engineer considerable time in examining system problems.

The wide range of TMDS hardware diagnostics available make it possible for varying degrees of analysis to take place on hardware subsystems while the system is in operation. Many diagnostic tests may impact the system adversely if run at peak load times. These tests can be run at lower use periods without adversely affecting system response.

And finally, TMDS's impact on the cold-load process is minimal. All TMDS processes are automatically started at system start-up time and contribute only 30 seconds to the total elapsed time needed to bring up the system.

TMDS is an important addition to Tandem's product line in that its benefits result in little cost to the user. It also provides valuable tools for diagnosing and maintaining hardware subsystems while having little impact on the performance of the system.

Reference

Blain, C., White, L., and Witte, W. 1987. Enhancements to TMDS. *Tandem Systems Review*. Vol. 3, No. 2. Tandem Computers Incorporated. Part no. 83940.

Acknowledgments
The author wishes to thank Stan Fong, Jim Troisi, and Wes Witte of TMDS development for their technical support and review.

Jim Mead joined Tandem in 1983. Initially in the Languages and Quality Assurance Group, he has also written test tools for QA. For the past year and a half he has worked in the performance area.

Overview of the C00 Software Release

The C00 and C-series releases concentrate on Tandem's continuing commitment to distributed transaction processing. In the C00 release, the emphasis is on management and connectivity in support of large system and large network environments.

Management

Three new products available in the C00 release allow users to manage the larger networks and systems made possible by Tandem's widening range of NonStop systems products. Distributed Systems Management (DSM), Remote Operations Facility (ROF), and Dynamic System Configuration (DSC), together with the previously available NetBatch™ product, provide a thorough management capability for Tandem NonStop systems and networks of NonStop systems.

Distributed Systems Management (DSM)

The DSM product is a critical component for large networks and systems; it allows remote management of all NonStop systems and associated networks. DSM also enables users to integrate the management of application systems and foreign hardware with the management of the Tandem systems and networks.

DSM provides four new products:

- VIEWPOINT™, a multifunction operations console application.
- Distributed Name Service (DNS), a facility to manage component names.
- Event Management System (EMS), a facility for managing and distributing event messages.
- Define Process (DP), a set of macro facilities for use with TACL (Tandem's Advanced Command Language) and VIEWPOINT in customizing the management environment.

Combined with two existing products, MEASURE™ system performance measurement tool and Network Statistics System (NSS), these new products provide a powerful management package.

Remote Operations Facility (ROF)

ROF allows users to start, stop, diagnose, and control NonStop VLX systems remotely even if they are not on the EXPAND network.

Dynamic System Configuration (DSC)

DSC allows users to add or modify controllers, communications lines, and devices that were not provided for in a SYSGEN, without having to stop the system. With DSC, users can adapt environments to meet changing network needs without affecting operations. This is particularly valuable for customers using Tandem's on-line transaction processing (OLTP) applications.

The Configuration Utility Program (COUP) can be used to add or delete devices and controllers, start or stop the I/O process for a device, alter the characteristics of a device or controller, and obtain information about the current configuration. Products implementing DSC in C00 are:

- TERMPROCESS (terminals).
- PRINTP (serial printers).
- IOPROCESS (parallel printers).
- TAPEPROCESS (all magnetic tape units).
- AM6520 device-specific access method.
- X25AM access method.
- ENVOY™ byte-oriented protocol.
- EXPAND networking feature of the GUARDIAN 90 operating system.

When using these communications protocols or I/O processes, characteristics of the following controllers can be modified on-line:

- 3401 universal interface.
- 6202 byte-synchronous controller.
- 6203 bit-synchronous controller.
- 6204 bit-synchronous controller.
- 6303 asynchronous controller.
- 32xx tape controllers (all).

More controllers and I/O processes will be added to the DSC list in future releases.

NetBatch

NetBatch job management system allows the user to schedule work remotely and to control work flows on large systems and large networks of systems. NetBatch became available in release B40.

Connectivity

In addition to management, C00 offers improvements in device connectivity, making it easier to bring transactions into the Tandem systems. Generalized Device Support (GDS) and PATHWAY enhancements, recently announced for both B40 and C00, provide this additional functionality.

Generalized Device Support (GDS)

GDS operates between the hardware (terminals, workstations, or other transaction capture equipment) and the user application. It permits the user to manage the protocol and presentation so that foreign devices can appear as standard device types (6530, 3270) to the application. This means that all the device-dependent code can be placed in one convenient module, allowing application programmers to concentrate on the applications. In addition, applications written for 6530 or 3270 can be used with virtually any device type without modification.

PATHWAY

The IDS feature of PATHWAY application programming language has been extended to permit better direct handling of intelligent devices when it is impractical or undesirable to develop a GDS environment. PATHWAY has also been improved to permit interrupt-driven processing through the new Unsolicited Message Processing (UMP) facility.

New Hardware Products

In addition to the new software products, the C00 release provides software support for the NonStop CLX low-cost full-function distributed system, the LASER-LX™ laser printer, and the super-high capacity 5200 Optical Storage Facility.

Underlying Architectural Changes

The addition of new products such as DSM and NetBatch has necessitated six significant changes to system software.

- The Subsystem Programmatic Interface (SPI) provides a standard, open architecture for management information exchanges.
- Event Management System (EMS) collects and routes event messages within a system and throughout a network.
- Completion codes are supplied by most Tandem products to control flows in batch jobs.
- GUARDIAN labeled tape handling was extended to most products that use tapes, providing a consistent user interface and more flexible tape management.
- Placing link control blocks (LCBs) in extended memory removes the threat of system failure because of a shortage of LCBs.
- Spool class defines provide for the control of printed output from batch jobs.

Subsystem Programmatic Interface (SPI)

SPI is the new standard for message formats and protocols used to exchange event and command information between operational subsystems and management applications within Tandem system networks.

SPI is the foundation for all DSM automated operations. It allows management information (such as event and command messages) to be defined in data formats easily handled by programs. This method replaces the textual message formats typically used for messages sent directly to human operators.

The programmatic nature of SPI messages facilitates the routing and processing of management information by Tandem's EMS, as well as by other management applications.

The SPI standard has been implemented in Tandem's key subsystems software products so that they may participate in DSM:

- SPI-formatted event messages generated by Tandem subsystems are collected by EMS for distribution to processing points throughout the network.
- SPI-formatted commands generated by automated operations programs can be received and executed by Tandem subsystems. (See Table 1.)

SPI is an open interface which can be invoked by programs written in TAL (the Tandem Application Language), COBOL85, and TACL—Command and Response (C&R) only. Besides integrating the management of business application programs into DSM, customers can use SPI to develop their own unique network and systems management applications that extend the DSM capabilities provided by Tandem.

- SPI program procedures allow users to easily generate event messages from their application programs. These application event messages can use the same DSM management services available to Tandem operational subsystems.
- SPI message formats used by Tandem subsystems for event and command messages are published so that users can develop automated management programs that process events or control the operations of these subsystems.
- Management information for devices or systems supplied by other manufacturers can be easily converted to SPI formats to integrate their management into the DSM environment.

SPI is not a product or a feature. It is an architecture for management information formats and protocols. The implementation of the SPI architecture throughout Tandem's software subsystems is the foundation that enables the DSM services and applications environment.

Event Management System (EMS)

EMS is a new feature of the GUARDIAN 90 operating system. It is available on all Tandem NonStop systems and provides a single facility for managing the collection and distribution of event information throughout a Tandem distributed systems network. EMS is the primary transport system for moving event information within a single system or across a Tandem systems network.

Table 1.
Subsystems that use SPI.

Subsystem	C&R*	EMS	Subsystem	C&R	EMS
$0, $Z0	X	X	GUARDIAN 90		X
DNS	X	X	LABELED TAPE		X
DP2		X	PATHWAY	X	X
EMS	X	X	PUP		X
EXPAND IOP	X	X	SCP	X	
EXPAND NCP		X	SNAX	X	X
FILE SYSTEM		X	TMDS		X
FOX	X	X	TMF	X	X
FUP	X		VIEWPOINT		X
GDS	X	X	X25AM	X	X

*C&R = Command and Response.

The EMS facility in each Tandem system collects event messages generated by any software subsystem residing in that system, logs these messages, and forwards the messages to appropriate management applications within the system.

In a multisystem network, individual EMS facilities can be linked together to move event information anywhere throughout the network.

EMS supports user-specified routing and distribution of event messages. EMS allows event messages to be routed to VIEWPOINT operations consoles for display or to other event processing applications. Users can determine EMS routing based on message type or the contents of each event message.

Completion Codes

Completion codes are passed by programs to the initiating process when they terminate. They are particularly useful in affecting the flow of steps within a NetBatch job. New facilities within C00 GUARDIAN 90 and TACL allow for the collection and retrieval of completion codes. Subsystems that have been modified in C00 to generate completion codes are as follows:

- Most language compilers. In addition, programs compiled in the following languages can be written to provide for returning completion codes: Ada, COBOL85, C, PASCAL, TAL, and TACL.
- Most utilities.

GUARDIAN 90 Tape Handling

TMF and BACKUP/RESTORE, which previously had their own tape handling routines, have been changed to use GUARDIAN 90 labeled tape services.

Relief from LCB Shortages

In C00, links may be carried out either with LCBs or with the new, extended-memory control block (XLB). XLBs, which are an almost unlimited resource, can alleviate instances of LCB shortages experienced by some users. XLBs are implemented on the following subsystems:

- File system.
- Tape process.
- EXPAND.

SPOOL Class Defines

SPOOL has been changed to recognize information from a Define, which specifies printer information. For example, forms information may be specified in a NetBatch job stream. Programs which have implemented SPOOL Class Defines include:

- Most languages (both compilers and run-time libraries).
- Most utilities and prompts.
- SPOOLER.

Conclusion

C00 provides new networking and connectivity options for the full range of NonStop systems, improving Tandem's ability to capture transactions and allowing improved use of our systems for OLTP.

Larry Marks is currently product manager for Communications Products. He has also served as manager for Network Management Product Development and chief of Operations and Technical Support for Tandem's 180-node computer network. He has worked in data processing and telecommunications for 20 years and has designed, implemented, and operated telecommunications networks throughout the United States, Europe, and Asia.

C00 Software Manuals

The Software Publications groups are pleased to announce the new and revised manuals and related parts for the C00 release. The manual packaging format has been changed to enable easier integration and use of manuals. Tabbed dividers are collated in place and binder spine stickers are packaged together with the manual. In addition, C00 manuals appear in a new grey design that conforms to the current corporate design standards.

Many of the manuals produced for the C00 release describe a new product family called Distributed Systems Management (DSM). Customers can use DSM to manage their systems interactively and to develop software that automates system management and operations tasks. The manuals include an introduction, descriptions of individual DSM components, and descriptions of the management interfaces for specific subsystems. Manuals throughout the library have been revised to include information related to DSM.

In addition to revisions of many existing manuals, System Software writers have added a new security administration guide, an optical disk manual, a new programmer's guide for the TACL command language, and two manuals for the NonStop CLX™ system. The system messages are now distributed across several manuals according to the type of message. The system management manual has its functions divided into two manuals: a system manager's guide and a system generation manual.

In the Data Management library, changes have been made to both the PATHWAY and Transaction Monitoring Facility (TMF) documentation. The *PATHWAY Application Programmer's Guide* provides conceptual SCREEN COBOL application programming discussions and guidelines. The *PATHWAY Management Programming Manual* and the *Transaction Monitoring Facility (TMF) Management Programming Manual* describe the programmatic interface for Distributed Systems Management (DSM). Two other manuals in the TMF library, the *Transaction Monitoring Facility (TMF) TMFCOM Reference Manual* and the *Transaction Monitoring Facility (TMF) Management and Operations Guide*, have been reorganized and rewritten.

In the Information Management Technology (IMT) library, a *PS TEXT EDIT and PS TEXT FORMAT User's Guide* and a two-volume *IMT Contact Guide* have been added. Volume 1 of the *Contact Guide* provides information for handling user problems with individual IMT products and for using TRANSFER ADMIN. Volume 2 includes all IMT and TRANSFER messages and recovery actions.

Table 1 lists the title, part number, and status of all the software manuals pertaining to the C00 software release. For more information about manual status, refer to the *Summary of Software Publications—Manuals and Related Products.*

Reference

Summary of Software Publications—Manuals and Related Products. Tandem Computers Incorporated. Part no. 84170.

Elise Levi joined Tandem in 1984, bringing with her six years of experience in writing, advertising, and technical publishing. She became manager of the Software Publications Production Editing and Graphics Coordination Group in 1986. Elise is now special projects manager for the Customer Education Group at Tandem.

Table 1.
C00 software manuals and related products.

Title	Part number	New or revised at C00
Data Management		
Data Definition Language (DDL) Reference Manual	84155	*
Data Management Reference Summary	82518	
ENABLE Reference Manual	82560	
ENABLE User's Guide	82571	
ENCORE User's Guide	82350	
ENFORM Reference Manual	82348	
ENFORM Reference Manual, Update 1	82194	
ENFORM Reference Manual, Update 2	82205	
ENFORM User's Guide	82349	
ENFORM User's Guide, Update 1	82195	
ENFORM User's Guide, Update 2	82206	
GUARDIAN 90 Utilities for NonStop SQL, Preliminary	82321	*
Introduction to ENFORM	82313	
Introduction to NonStop SQL	82317	
Introduction to PATHMAKER	84070	
Introduction to PATHWAY	82339	
Introduction to the Transaction Monitoring Facility (TMF)	82528	
Introduction to TRANSFER Delivery System	82323	
NonStop SQL Benchmark Workbook	84160	*
NonStop SQL Conversational Interface Reference Manual, Preliminary	82319	*
NonStop SQL Messages Manual, Preliminary	82329	*
NonStop SQL Programming Reference Manual, Preliminary	82318	*
NonStop SQL Quick Start, Preliminary	82320	*
PATHAID Reference Manual	82428	
PATHMAKER DB Requester Template for the 6530 Terminal	84121	
PATHMAKER DB Requester Template for the 6526 Terminal	84122	
PATHMAKER Programming Manual	82450	
PATHMAKER and Screen Painter Template for the 6530 Terminal	84009	
PATHMAKER and Screen Painter Template for the 6526 Terminal	84018	
PATHWAY Application Programmer's Guide	84143	*
PATHWAY PATHCOM Reference Manual	84078	*
PATHWAY Management Programming Manual, Volume 1	84112	*
PATHWAY Management Programming Manual, Volume 2	84113	*
PATHWAY SCREEN COBOL Reference Card	84075	*
PATHWAY SCREEN COBOL Reference Manual	84072	*
PATHWAY SCREEN COBOL Utility Program (SCUP) Reference Card	84158	*
PATHWAY SCREEN COBOL Utility Program (SCUP) Reference Manual	84071	*
Remote Duplicate Database Facility (RDF) Reference Card	84081	
Remote Duplicate Database Facility (RDF) System Management Manual	84077	
Transaction Monitoring Facility (TMF) TMFCOM Reference Card	84059	*
Transaction Monitoring Facility (TMF) TMFCOM Reference Manual	84067	*
Transaction Monitoring Facility (TMF) Management Programming Manual	84065	
Transaction Monitoring Facility (TMF) Management and Operations Guide	84069	*

Continued next page

Table 1.
Continued

Title	Part number	New or revised at C00
Information Management Technology		
EDIT Sleeve and Binder	84145	*
EDIT User's Guide and Reference Manual	84082	*
FAXLINK Cover Sheets	82558	
FAXLINK Reference Manual	82545	
FAXLINK Sleeve and Slipcase	84008	
FAXLINK User's Guide	82574	
IMT Contact Guide Sleeve and Binder Volume 1: Reference Manual	84100	*
IMT Contact Guide Sleeve and Binder Volume 2: Messages Manual	84098	*
IMT Contact Guide, Volume 1: Reference Manual	84085	*
IMT Contact Guide, Volume 2: Messages Manual	84087	*
MULTILAN User's Guide	84058	
MULTILAN Sleeve and Binder	84101	
PC LINK PC6530 EDIT/VS Template for the IBM PC	44701	
PC LINK PC6530 Emulation Template for the IBM PC (Rectangle)	82535	
PC LINK PC6530 Reference Manual for the IBM PC	82692	
PS MAIL Sleeve and Binder for TTY Terminals	84001	
PS MAIL Quick Start for 3270 Terminals	84032	
PS MAIL Quick Start for 6530 Terminals	84033	
PS MAIL Quick Start for TTY Terminals	84034	
PS MAIL Reference Card for TTY Terminals	84035	
PS MAIL Reference Manual for TTY Terminals	84036	
PS MAIL Sleeve and Slipcase	84002	
PS MAIL User's Guide and Reference Manual for 3270 Terminals	84042	
PS MAIL User's Guide and Reference Manual for 6530 Terminals	84043	
PS MAIL User's Guide for TTY Terminals	84044	
PS TEXT EDIT Sleeve and Binder	84000	
PS TEXT EDIT—EDIT/VS to TEDIT Reference Card	82479	
PS TEXT EDIT Reference Card	84096	*
PS TEXT EDIT Reference Manual	82550	
PS TEXT EDIT Reference Manual, Update 1	82242	*
PS TEXT EDIT (3270)/PS MAIL (3270) Template for the 3270 Terminal (Rectangle)	84038	
PS TEXT EDIT (3270)/PS MAIL (3270) Template and Stickers for the 3270 Terminal	84037	
PS TEXT EDIT (6530)/PS MAIL (6530) Template for the DYNAMITE Terminal	84039	
PS TEXT EDIT (6530)/PS MAIL (6530) Template for the IBM PC	84040	
PS TEXT EDIT (6530)/PS MAIL (6530) Template for the 6530 Terminal	84041	
PS TEXT EDIT (6530)/PS MAIL (6530) Template for the 6526 Terminal	82498	
PS TEXT EDIT (6530)/PS MAIL (6530) Template for the Tandem 6AX Workstation	84057	
PS TEXT EDIT Quick Start for 3270 Terminals	82567	
PS TEXT EDIT Quick Start for 6530 Terminals	82595	
PS TEXT EDIT and PS TEXT FORMAT	84097	*
PS TEXT EDIT and PS TEXT FORMAT User's Guide	84084	*
PS TEXT FORMAT Sleeve and Binder	84003	
PS TEXT FORMAT Quick Start	82566	
PS TEXT FORMAT Reference Card	82572	
PS TEXT FORMAT Reference Manual	82478	
PS TEXT FORMAT Reference Manual, Update 1	82243	*
PS TEXT FORMAT to TGAL Reference Card	82573	
TRANSFER Installation and Management Guide, Preliminary	82492	
TRANSFER Programming Guide, Preliminary	84030	
WPLINK Batch Gateway Reference Card	82472	
WPLINK Batch Gateway Reference Manual	82471	
WPLINK Sleeve and Slipcase	84004	
WPLINK Translation Reference Manual	82494	
WPLINK User's Guide for the IBM Displaywriter	82476	
WPLINK User's Guide for the Wang OIS Workstation	82474	
WPLINK User's Guide for the Wang VS Workstation	82475	

Continued next page

Table 1.
Continued

Title	Part number	New or revised at C00
Languages		
Ada User's Guide	82523	*
ANSI Reference Manual for the Ada Programming Language	84063	*
C Reference Manual	84016	
C Reference Summary	82537	
COBOL Pocket Guide	82575	
COBOL Reference Manual, Volume 1	82589	
COBOL Reference Manual, Volume 2	82590	
COBOL User's Guide	82389	
COBOL85 Reference Manual, Volume 1	82520	
COBOL85 Reference Manual, Volume 2	82521	
COBOL85 Reference Manual, Update 1	82239	*
COBOL85 Reference Summary	82538	
EXTENDED BASIC Pocket Guide	82379	
EXTENDED BASIC Reference Manual	82580	
EXTENDED BASIC User's Guide	82385	
FORTRAN Pocket Guide	82577	
FORTRAN Reference Manual	82515	
MUMPS Pocket Guide	82578	
MUMPS Reference Manual	82542	
Pascal Reference Manual	82510	
Transaction Application Language (TAL) Pocket Guide	82376	
Transaction Application Language (TAL) Reference Manual	82485	
Transaction Application Language (TAL) Reference Manual, Update 1	82240	*
Networking and Data Communications		
6100 ADCCP Programming Manual	82411	
6100 ADCCP Programming Manual, Update 1	82227	
6100 BSC Programming Manual	82412	
6100 MPS-B Programming Manual	82413	
6100 MPS-TINET Programming Manual	82414	
CP6100 I/O Process Programming Manual	82410	
CP6100 I/O Process Programming Manual, Update 1	82225	
Asynchronous Terminal and Printer Processes Programming Manual	84105	
Asynchronous Terminal and Printer Processes Programming Manual, Update 1	82237	*
BELLEVUE Frame Editor User's Guide	82448	
BELLEVUE Programmer's Guide	84062	
BELLEVUE Subsystem Manager's Guide	82480	
Communications Management Interface (CMI) Operator's Guide	84056	
Communications Management Programming Manual	84110	*
Communications Utility Program (CUP) Reference Manual	82430	
Device-Specific Access Method—AM6520	82433	
Device-Specific Access Methods—AM3270/TR3271	82451	
ENVOY Byte-Oriented Protocols Reference Manual	82582	
ENVOYACP Bit-Oriented Protocols Reference Manual	82588	
ENVOYACP/XF Bit-Oriented Protocols with Extended Functions Reference Manual	84116	*
EXCHANGE Reference Manual	82568	
EXCHANGE Reference Manual, Update 1	82220	
EXPAND Management Programming Manual	84109	*
EXPAND Reference Manual	82452	
EXPAND Reference Manual, Update 1	82230	
FOX Management Programming Manual	84136	*
GDS Management Programming Manual	84164	*
General Device Support (GDS) Programming Manual	84129	*
Introduction to SNAX	82466	
Introduction to the Tandem 6100 Communications Subsystem	84064	
Introduction to Tandem Data Communications	82511	
MULTILAN Manager's Guide	82489	
MULTILAN Programmer's Guide	82488	
OSI Layer 4 Access Method (OSI4AM) Subsystem Management amd Programming Manual	82447	
PTRACE Reference Manual	84051	
SAFE-T-NET 3501 Cryptographic Subsystem Operation and Programming Manual	82446	
SNAX/CNM Manual	84133	*
SNAX Configuration and Control Manual	84123	*
SNAX Device-Access Methods Programming Manual	82469	
SNAX/HLS Manual	84125	*
SNAX Management Programming Manual	84124	*
SNAX Reference Summary	82591	
Subsystem Control Facility (SCF) Reference Manual	84159	*
Tandem-to-IBM Link Reference Manual for IBM Users	82050	
Tandem-to-IBM Link Reference Manual for Tandem Users	82055	
Tandem Hyper Link (THL) Reference Manual	82056	
X25AM Management Programming Manual	84135	*
X.25 Access Method (X25AM) Programming Manual	82431	

Continued next page

Table 1.
Continued

Title	Part number	New or revised at C00
System Software		
5200 Optical Storage Facility (OSF) Reference Manual	84115	
Distributed Name Service (DNS) Manual	84093	*
Distributed Systems Management (DSM) Master Index	11183	*
Distributed Systems Management (DSM) Programming Manual	82587	*
DP1-DP2 File Conversion Manual	82407	
DP1-DP2 File Conversion Manual, Update 1	82232	
Dynamic System Configuration Manual	84167	*
ENSCRIBE Programmer's Guide	82594	
Event Management System (EMS) Manual	84092	*
GUARDIAN 90 Operating System Event Messages Manual	84127	*
GUARDIAN 90 Operating System Operations Reference Summary	84271	*
GUARDIAN 90 Operating System Programmer's Guide	84083	*
GUARDIAN 90 Operating System Programming Reference Summary	84089	*
GUARDIAN 90 Operating System User's Guide	84132	*
GUARDIAN 90 Operating System Utilities Programming Manual	84128	*
GUARDIAN 90 Operating System Utilities Reference Manual, Volume 1	84138	*
GUARDIAN 90 Operating System Utilities Reference Manual, Volume 2	84139	*
GUARDIAN 90 Operating System Utilities Reference Manual, Volume 3	84140	*
GUARDIAN 90 Operating System Utilities Reference Manual, Volume 4	84250	*
Introduction to Distributed Systems Management (DSM)	84091	*
MEASURE Reference Manual	84156	*
MEASURE User's Guide	84157	*
NetBatch Manual	84074	*
NonStop CLX Help Booklet	84166	*
NonStop CLX Local Operations Manual	84126	*
NonStop CLX Systems Management Manual	84114	*
NonStop EXT Systems Operator's Guide	82490	
NonStop VLX System Operator's Guide	84080	*
Operator Messages: Console Format	84076	*
Operator Messages: Distributed Systems Management (DSM) Display Format	84103	*
Remote Operations Facility (ROF) User's Guide	84163	*
SAFEGUARD Reference Card	84014	
SAFEGUARD Reference Manual	84161	*
SAFEGUARD User's Guide	84162	*
Security Administration Guide	84134	*
Spooler Programmer's Guide	82394	
System Description Manual	84017	
System Generation Manual, Volume 1	84094	*
System Generation Manual, Volume 2	84095	*
System Manager's Guide	84117	*
System Operator's Guide	84079	*
System Procedure Calls Reference Manual, Volume 1	84118	*
System Procedure Calls Reference Manual, Volume 2	84119	*
System Procedure Errors and Messages Manual	84141	*
TACL Programmer's Guide	84111	*
Tandem Maintenance and Diagnostic System (TMDS) Reference Manual	82386	
VIEWPOINT Manual	82592	*
VIEWPOINT Template for the 6530 Terminal	11121	*
VIEWPOINT Template for the Tandem 6AX Workstation	11124	*
Tools and Utilities		
BINDER Manual	84104	*
BINDER Reference Card	84169	*
CROSSREF Manual	82597	*
DEBUG Manual	82598	
ENTRY Screen Formatter Operating and Programming Manual	82020	
ENTRY520 Screen Formatter Operating and Programming Manual	82053	
FASTSORT Manual	84144	*
INSPECT Manual	84149	*
INSPECT Reference Summary	84265	*
Labeled Tapes User's Guide for COBOL	82422	
TGAL Manual	82526	
Tools and Utilities Pocket Guide	82304	

Continued next page

Table 1.
Continued

Title	Part number	New or revised at C00
Small Systems (Austin)		
LXN/System-Level Hardware Binder and Slipcase	41270	
Introduction to LXN	83800	
LXN Installation Guide	37490	
LXN Installation Guide Supplement	33790	
LXN Maintenance Manual	83802	
LXN Technical Reference Manual	83803	
LXN Multibus Adapter Board Installation Manual	83805	
LXN/UNIX Binder and Slipcase	41271	
UNIX System V Administrator Guide and Reference Manual	37780	
UNIX System V Administrator Guide and Reference Manual Update Package	48324	
UNIX System V User's Guide	83806	
UNIX System V Commands Directory	37779	
UNIX System V Commands Directory Update Package	48323	
UNIX System V User's Reference Manual	37781	
UNIX System V User Reference Update Package	48325	
UNIX System V Programming Guide	37782	
UNIX System V Programmer and Error Message Reference Manual	37783	
UNIX System V Support Tools Guide	83812	
LXN/UNIX Documenter's Workbench Binder and Slipcase	41271	
UNIX System V Documenter's Workbench, Volume 1	83813	
UNIX System V Documenter's Workbench, Volume 2	83814	
LXN/Communications Binder and Slipcase	41272	
GUARDIAN/UNIX File Transfer User's Guide	37786	
GUARDIAN/UNIX X.25 File Transfer User's Guide	48232	
LXN LAN Introduction	37892	
LXN LAN ENET Reference Manual	37893	
LXN LAN Microsoft Networks User's Guide	37894	
LXN LAN Microsoft Networks Manager's Guide	37895	
LXN LAN PC-Interface Administrator's Guide	37899	
LXN LAN PC-Interface Programmer's Guide	37900	
LXN LAN PC-Interface User's Guide	37898	
LXN X.25 Interface Reference Guide	37784	
LXN SNA System Administration and Host Support Guide	37785	
LXN SNA User's Guide	83816	
LXN/Database Binder and Slipcase	41273	
Informix 4GL Reference Manual Volume One	83825	
Informix 4GL Reference Manual Volume Two	83826	
Informix 4GL User Guide	83824	
Informix ESQL/C Programmer's Manual	83822	
Informix-SQL Reference Manual	83820	
Informix-SQL User Guide	83819	
LXN/Languages Binder and Slipcase	41274	
LPI-BASIC Reference Manual	38030	
LPI-BASIC User's Guide	38029	
Micro Focus Animator Operating Guide	83831	
Micro Focus Level II COBOL/ET Language Reference Manual	83828	
Micro Focus Forms-2 Utility Manual	83830	
Micro Focus Level II COBOL/ET Operating Guide	83827	
Micro Focus Upgrade III Operating Guide	83829	
Professional Pascal Language Extensions	83833	
Professional Pascal Programmer's Guide	83832	
Professional Pascal Programmer's Primer	36895	
SVS FORTRAN Reference Manual	38031	

Continued next page

Table 1.
Concluded

Title	Part number	New or revised at C00
Terminals (Austin)		
6526 Bar Code Terminal Installation and Operation Guide	82735	
6526 Multi-Page Terminal Installation and Operation Guide	82714	
6526 Multi-Page Terminal Programmer's Guide	37704	
653X Integrated Options—Installation and Operation Guide	82677	
653X Multi-Page Terminal Installation and Operation Guide	82308	
653X Multi-Page Terminal Programmer's Guide	82310	
653X Terminal Emulator Programmer's Guide	82659	
6AI Alternate Input Device Option for 6530 Terminal Installation and Operation Guide	82652	
6VI Voice Input Option for 653X Terminal Installation and Operation Guide	82671	
EM3270 Keyboard Template for 653X Terminals	42646	
EM3270 Keycap Stickers for 654X and 6535/36/37	47507	
EM3270 Option for 653X Family User's Guide	82668	
Getting to Know T-TEXT	82655	
Printer Interface Option for 653X Terminals Installation and Operation Guide	82650	
T-TEXT User's Manual	82656	
Video Monitor Interface Option for 6530 Terminal Installation and Operation Guide	82653	
Word Processing Option for 653X Terminals Installation and Operation Guide	82654	
Workstations and Printers (Austin)		
6AX Workstation Binder and Slipcase	48082	
6AX Operations Guide	82701	
6AX MS-DOS User's Guide	48268	
6AX GW-BASIC User's Guide	48269	
6AX PC6530 User's Guide	34220	
6AX Maintenance Manual	82702	
6AX Configuration Guide	82727	
6AX Hardware Technical Reference	82709	
654X AMT6530 Terminal Emulator User's Guide	82688	
654X CCS Programmer's Guide	82693	
654X GW-BASIC User's Guide	82664	
654X IXF User's Guide	82669	
654X Macro Assembler Programmer's Guide	82662	
654X Maintenance Manual	82657	
654X MS-DOS Programmer's Guide	82660	
654X MS-DOS User's Guide	82661	
654X National Language Supplement	82691	
654X Operations Guide	82658	
654X Technical Reference	82665	
Blank Template for 654X and 6535/36/37	47172	
The DYNAMITE Configuration Guide	82703	
EM3270 Keyboard Template for 654X and 6535/36/37	47484	
ENABLE Template for 654X and 6535/36/37	47171	
VS Template for 654X and 6535/36/37	47170	
Communication Control System Programmer's Guide	37271	
PC6530 User's Guide	34219	
EM6530PC Terminal Emulator Programmer's Guide	37168	
PCFORMAT User's Guide	82679	
Model 6820 Terminal Cluster Concentrator (TCC) Installation and Operation Guide	82651	
Model 554X Serial Matrix Printer Installation and Operation Guide	82678	
Model 6600 Intelligent Cluster Controller Installation and Operation Guide	82684	
Supplemental Publications		
1¼" Binder, Custom 9" × 11"	84257	*
1½" Binder, Custom 8" × 9"	84147	*
2" Binder, Custom 9" × 11"	84146	*
A06-B40 Binder Spine Labels for Gray Binders	11141	*
Catalog of Software Publications and Related Products	82552	
C00 Release Software Documents (Softdocs)	84420	*
Introduction to Tandem Computer Systems	82503	
Master Index for the NonStop Systems	82586	
Summary of Software Publications—Manuals and Related Products	84170	*

The following are trademarks of Tandem Computers Incorporated: BINDER, CLX, CROSSREF, ENABLE, ENCORE, ENFORM, ENSCRIBE, ENTRY, ENVOY, EXCHANGE, EXPAND, EXT, FASTSORT, FAXLINK, FOX, GUARDIAN, GUARDIAN 90, INSPECT, LXN, MEASURE, MULTILAN, NetBatch, NonStop, NonStop SQL, PATHMAKER, PCFORMAT, PC LINK, PS MAIL, PS TEXT EDIT, PS TEXT FORMAT, SAFEGUARD, SAFE-T-NET, 6AX, TACL, TAL, Tandem, TGAL, THL, TIL, TMDS, TMF, TRANSFER, T-TEXT, VIEWPOINT, VLX, and WPLINK. BELLEVUE is a trademark of Tandem Computers available only to users of the Bildschirmtext (BTX) videotex network.

Ada is a registered trademark of the U.S. Government (Ada Joint Program Office). IBM and IBM Displaywriter are trademarks of International Business Machines Corporation. Wang, Wang OIS, and Wang VS are trademarks of Wang Laboratories, Inc. UNIX is a trademark of AT&T Bell Laboratories. Microsoft, MS-DOS, and GW-BASIC are trademarks of Microsoft Corporation. Informix is a trademark of INFORMIX Software, Inc.

The Tandem Corporate Education Group has added six courses to its software education curriculum. The new courses are described in this article. Following the descriptions is a list of Tandem's Customer Education Centers.

As more new courses are offered throughout the year, information about them will be made available. To obtain course information, enroll, or order any self-paced course, contact your Tandem sales representative or your nearest Tandem Education Center. Refer to the *Tandem Software Education Catalog and Schedule* (October 1987) for full descriptions and schedules of all Tandem classes.

Introduction to NonStop SQL

This 4½-day course is a general introduction to the NonStop SQL product and serves as a prerequisite to more advanced NonStop SQL courses. Lab sessions supplement classroom lectures and provide hands-on practice.

The Introduction to NonStop SQL course includes the concepts of creating a SQL database, selecting and updating information on a SQL database, generating SQL reports, and compiling and executing COBOL85 programs that contain SQL statements.

Audience

The course is designed for application designers, database administrators, Tandem analysts (presales, account, and support), i.e., anyone requiring a general introduction to the NonStop SQL product.

Prerequisites

A general Tandem background is required. Experience with PATHWAY, TMF, and ENSCRIBE, and knowledge of logical database design are helpful.

PATHWAY System Management (38802)

This course covers the skills and information needed to manage PATHWAY, Tandem's transaction processing system. Presentations, demonstrations, and laboratory exercises focus on configuring, changing, monitoring, and maintaining PATHWAY systems.

Topics include:

- Tandem system architecture overview.
- Configuring and starting PATHWAY systems.
- The PATHWAY set commands.
- TCP2 configuration specifics.
- System maintenance commands.

Audience

PATHWAY system managers and system administrators.

Prerequisite

Concepts and Facilities (or equivalent) is the only prerequisite.

Fast-Track PATHWAY (38920)

The Fast-Track PATHWAY course accelerates the training of application programmers. The class trains experienced COBOL programmers to program for the Tandem system in ten days—half the time formerly required. The focus of the class is toward learning to program in the Tandem PATHWAY environment.

This course includes:

- Concepts of Tandem hardware and the GUARDIAN operating system.
- The utilities used in the development cycle.
- ENSCRIBE file structures.
- The Data Definition Language (DDL) and PATHWAY products.
- PATHWAY configuration commands.
- The SCREEN COBOL language.
- Interprocess communication.
- On-line application design considerations.
- An introduction to PATHMAKER™.

Audience
Experienced COBOL programmers.

Prerequisites

- Concepts and Facilities.
- Introduction to Tandem Application Design.
- PATHWAY for Programmers.

PATHMAKER (38820)

This course focuses on implementation of a PATHWAY application in a lab/lecture environment. Additional modules cover installation for system managers and design considerations for designers and database administrators.

Major topics covered in this class are:

- Generating requesters.
- Screen painters.
- Coding services.
- Generating servers.
- Generating a test PATHWAY application system.
- PATHMAKER installation.

Audience
Application programmers, system managers, application designers, and database administrators who will be involved in using PATHMAKER to generate PATHWAY applications.

Prerequisites

- At least one person experienced in PATHWAY system management and PATHWAY application at each PATHMAKER site.
- Concepts and Facilities.
- Introduction to COBOL Programming or Advanced COBOL Programming and Tandem Extensions.
- PATHWAY or PATHWAY for Programmers.

Designing for Performance: Basic Concepts (39457) (*Customer Support Video Series*)

This is the first videotape in the Customer Support Video Series. Designing for Performance: Basic Concepts discusses how business applications can be managed to meet goals such as response time, throughput, and growth. The tape treats management in terms of goals, control, understanding, and required information. Topics include:

- Tandem architecture as it applies to application modelling.
- Application of theories to modelling and design efforts.
- Performance terms and concepts.

Audience

Any Tandem customer in the applications design, system management, or operations area.

Prerequisites

- PATHWAY (or equivalent).
- GUARDIAN Internals for NonStop Systems (recommended).
- Introduction to Tandem Application Design (recommended).
- MEASURE Seminar (recommended).

Introduction to Local Area Networks (38905)

This computer-based training course presents the basic terminology and concepts of local area networks (LANs) using text, graphics, and animation on the 6AX or IBM PC/XT/AT or compatible workstations. A brief overview of LANs is provided with in-depth coverage of topologies, media, and access methods.

Audience

Anyone interested in learning the basic terminology and concepts of LANs.

Prerequisites

None

Jack Limper recently moved from Software Education Course Development to Support Program Management. He joined Tandem in 1984 after more than 14 years with another major computer vendor. Currently he manages the Application Course Development Group.

Tandem Customer Education Centers

United States

Contact the following education centers to enroll at any domestic location.

California
5901 Green Valley Circle
Suite 400
Culver City, CA 90230
(213) 417-3922

39899 Balentine Drive
Suite 230
Newark, CA 94560
(415) 490-7353

Illinois
500 Park Blvd.
Suite 1400
Itasca, IL 60143
(312) 773-1750

Michigan
21800 Haggerty Road
Farmington Hills, MI 48018
(313) 344-0200

New Jersey
777 Terrace Avenue
Second Floor
Hasbrouck Heights, NJ 07604
(201) 288-6050

Texas
4001 McEwen
Suite 300
Dallas, TX 75244
(214) 980-0311

Washington, D.C.
12100 Sunrise Valley Drive
Reston, VA 22091
(703) 476-3222

Classes are also held at the following locations:

Georgia
150 Interstate North Parkway
Suite 100
Atlanta, GA 30339

Pennsylvania
Commerce Court Office Building
4 Station Square
Suite 740
Pittsburgh, PA 15219

International

Belgium
Tandem Computers N.V./S.A.
Excelsiorlann 37
B-1930 Zaventem
02-2-7209588

Canada
Tandem Computers Canada LTD
2000 McGill College Avenue
Suite 800
Montreal, Quebec H3A 3H3
(514) 282-1488

Tandem Computers Canada LTD
7270 Woodbine Avenue
Third Floor
Markham, Ontario L3R-4B9
(416) 475-8222

Denmark
Tandem Computers A/S
Postbox 171
H.J. HostT Vej 3
DK-2600 Glostrup
05-1-720055

England
Tandem Computers Europe Inc.
Tandem House, Mendy Street
High Wycombe
Buckinghamshire HP112NZ
0494-21277

France
Tandem Computers S.A.
163 Avenue Charles de Gaulle
92200 Neuilly-sur-Seine, Paris
331-4738-1196

Germany
Tandem Computers GmbH
Geschaeftsstelle Duesseldorf
Heinrich-Hertz-Strasse 2
4010 Hilden
02103-572-0

Tandem Computers GmbH
Niederlassung Frankfurt
Berner Strass 34
6000 Frankfurt/Main 56
069-50003-80

Hong Kong
Tandem Computers Hong Kong LTD
448 Wing On Plaza
Tsim Sha Tsui East, Kowloon
3-7218136

Italy
Tandem Computers Italia S.P.A.
Viale del Ghisallo, 20
20151 Milano
02-301-2615

Japan
Tandem Computers Japan LTD
Yasukuni Kudan Minimi Building
2-13-14 Kudan Minami, Chiyoda-Ku
Third Floor
Tokyo 102
03-237-9531

Netherlands
Tandem Computers B.V.
"Plus Point"
Jupiterstraat 146-3
2132 HG Hoofddorp
02503-68787

Norway
Tandem Computers Norway A.S
O. H. Bangsvei 51
P.O. Box 94
N-1322 Hoevik
02-123330

Singapore
Tandem Computers International
24 Raffles Place
19-03 Clifford Centre
Republic of Singapore 0104
05-533-7611

Sweden
Tandem Computers AB
Norgegatan 1
Box 1254
S-163 13 Spaanga
08-750-7540

Switzerland
Tandem Computers A.G.
Zweierstrass 138, Postfach
CH-8036 Zurich
01-461-3025

Distributors

Training is also available from the following Tandem distributors:

Finland
Oy DAVA AB
Porkkalankatu 5
SF-00180 Helsinki
3580-1241

Mexico
Tandem Computers de Mexico, S.A. de C.V.
Moras No. 313
Col. del Valle
03100 Mexico, D.F
(905) 559-9177

Taiwan
Syscom Computer Engineering Co
53 Jen Ai Road, Sec. 3
Ninth Floor
Taipei, Taiwan, R.O.C.
02-773-1302

Technical Paper: The Role of Optical Storage in Information Processing

The data processing industry demands higher-capacity, lower-cost storage to replace paper cost-effectively and bring increasing amounts of data into the computer environment. Optical disk technology responds to this need and challenges systems architects and integrators to identify and develop successful applications for storage devices using this technology. To do so, they must answer the following questions:

- What storage functions can optical disk perform more efficiently and cost-effectively than existing products?
- What new storage applications are made feasible by optical disk?

The key to determining profitable product applications lies in understanding the basic theory of operation, the performance characteristics, the economics, and the standards status of the various optical disk technologies. The new Tandem 5200 Optical Storage Facility, discussed later in this article, brings the advantages of optical memory to the mainframe computer environment and provides online archive benefits to Tandem's customers.

Optical Disk Characteristics

Optical disks provide large capacity, low-cost random-access storage on removable media.

Two distinct types of optical disk are currently available. Compact disk read-only memory (CDROM) is a read-only medium from the user's point of view. Write once read many (WORM) optical disk may be written once and read repeatedly by the user. Unlike magnetic media, WORM media cannot be rewritten, but in many applications, that might be more of an asset than a liability.

This article appeared in a slightly different form in the June 1987 edition of *Computer Technology Review.*

Optical disks are composed of a substrate layer of polycarbonate or glass, a reflective layer, and a transparent protective layer. Tellurium oxides are most frequently used for the reflective layer. Figure 1 illustrates the structure of an optical disk. Data is on a continuous spiral groove, with each complete revolution considered a track. Some WORM drives store data in the area between the grooves called the *land*.

Virtually Error-Free

The raw-bit error rate of the media is between one in 10**4 and one in 10**6. Reed-Solomon error correcting codes (ECC) reduce the error rate to one in 10**12, a level acceptable for digital data storage. Because of the high capacity of the media, the 15% overhead for these correction schemes does not represent a significant burden.

Longevity

Optical disk media are also characterized by very long life. Media life is defined as the period of time over which the manufacturer guarantees the specified correctable error rate for reads. CDROM lifetimes are specified at 30 years. WORM media lifetimes are conservatively specified at 10 years, although one vendor has committed to a 30-year lifetime specification based on accelerated life stress test results.

Disk Rotation Methods

Optical disk uses two methods of rotation. In one method, constant linear velocity (CLV) maximizes disk capacity at the expense of access time. The speed of rotation is inversely proportional to the track diameter. Some delay is incurred in changing rotational velocity during track-to-track access.

Figure 1

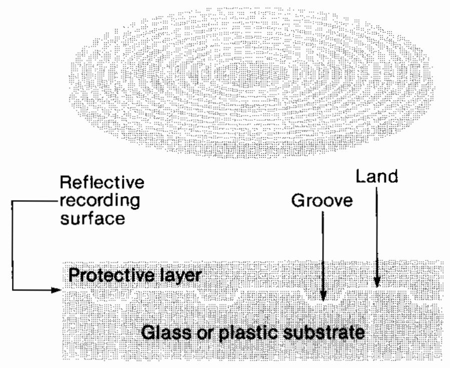
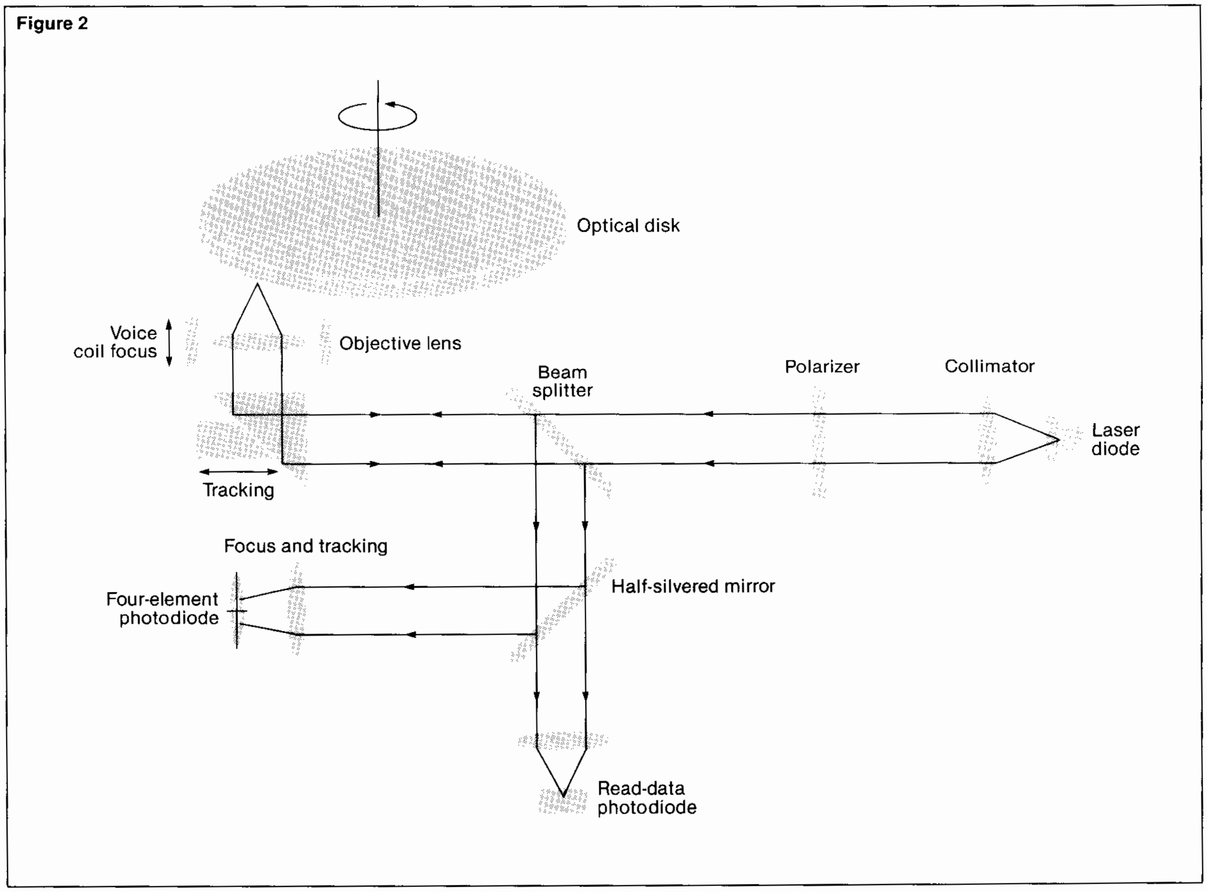


Figure 1.

Optical disk structure. Data may be stored on concentric tracks or on a continuous spiral groove. Each complete revolution is considered a track.

Constant angular velocity (CAV) sacrifices disk capacity for faster access; no time is spent adjusting the spin speed during random access. CLV capacity is typically 50% greater than CAV capacity. Preliminary specifications for 5¼-inch WORM access times are 150 milliseconds for CLV drives compared with 50 milliseconds for CAV drives.

Figure 2.
Optical read/write head and disk.



A Common Read Mechanism

The read mechanism is similar for all types of optical disk memory. A laser beam is focused on the track by servo mechanisms that position and focus the beam. The reflected pattern contains the encoded stored data. Figure 2 is a diagram of an optical disk read/write head showing the laser, lens system, photodiode, and disk.

Gallium aluminum arsenide (GaAlAs) semiconductor lasers are used in commercial optical disk drives. Their size, high reliability, efficiency, and low cost are essential in making optical disk technology viable. The laser produces an 830-nanometer wavelength light. The diameter of the pits storing the data is nearly the same as that of the wavelength. Objective lenses carefully gather the light into a pit-sized spot.

The lens system used by optical disks has two advantages over magnetic disk. With magnetic disk, the read/write head is moved closer to the disk to increase bit density, and state-of-the-art magnetic heads fly as close to the disk surface as 0.4 micrometer. In contrast, the optical head is 2000 times as far from the disk because bit density is provided by focus instead of proximity. This distance removes the threat of head crashes and facilitates removability of media.

High Capacity

Optical disk memory is characterized by extremely high capacity; bit densities of 3 × 10**8 bits per square inch are 20 times the density of state-of-the-art Winchester magnetic disks. High capacity is a major factor in decreased storage costs. Removability is an essential attribute for media that can either be read only (CDROM) or written once (WORM). Basic concepts of magnetic recording are discussed in a previous issue of the *Tandem Systems Review* (Ng, 1986).

The second advantage is that optical disks are covered by a transparent protective coating through which the light is focused. Thus fingerprints and dust on the coating are merely blurs; the steeply converging laser focuses through them, and they do not cause the problems they would for a magnetic disk. Figure 3 illustrates the inherent media protection of optical disks.

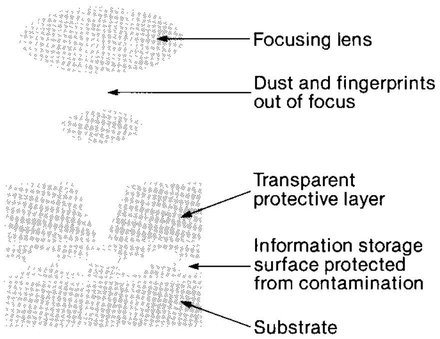
Data is read by measuring the light reflected back through the objective lens. Most of the light hitting a pit is transmitted, while light hitting an unwritten area is reflected. The photodiode produces a signal that varies with the intensity of the reflected light. This modulated signal contains the stored information and is converted to digital data.

Applications

Videodisk was the first optical disk product application. This product was eclipsed by videotape because the tape allows the viewer to record from broadcast programming, watch the program, and recycle the tape. Videodisk systems are play only. Designers learned a hard lesson from videodisk; they must first carefully assess users' needs and wishes.

Audio compact disk, or CD, was the second major optical disk product. The medium is impervious to wear, an essential attribute for recordings played repeatedly. And, in contrast to the battle between VHS and Beta videotape formats, which confused the consumer and delayed market growth, CD pioneers Sony and Phillips agreed upon the disk size, storage format, and playback mechanics so that all CDs play in any drive.

Figure 3



Focusing lens
Dust and fingerprints out of focus
Transparent protective layer
Information storage surface protected from contamination
Substrate

Figure 3.
Inherent media protection of optical disks. Minimization of handling damage is desirable in removable media.

Because CDs store data in digital form, they are suitable for storing digital data for computer applications. A CDROM is physically identical to an audio CD. The audio CD player reads the digital data and uses digital-to-analog conversion to produce sound. By deleting the conversion circuitry and creating a computer interface for drive control and data transfer, the audio CD player design can be adapted to computer applications. CDROM drives using the small computer system interface (SCSI) and IBM PC-compatible interfaces are available.

The only difference between audio CD and CDROM is in the organization of data within the user data area. CDROM specifies a more powerful ECC and more explicit absolute block addressing. Consequently, CDROM disks can be manufactured by the same factories, producing significant economies of scale and avoiding the costs of starting a new manufacturing process.

CDROM Format

The CDROM format, known as the *Yellow Book*, was developed by Sony and Phillips. The popular 5¼-inch full- and half-height form factors easily accommodate the 120-mm disk size, thus facilitating integration of CDROM drives with PCs. The Yellow Book standard also defines two physical block formats for data storage. Depending upon the selected playback option, 550 or 600 Mbytes of user data can be stored on disk. CDROM uses CLV drives with average access times between 500 and 2000 milliseconds and transfers data at a 1.3-Mbit-per-second rate.

Although the Yellow Book defines the physical layout of data on the disk, ECC, and encoding, it does not specify a logical file structure or a directory format. The lack of standards impeded early efforts to use CDROM as a publishing medium.

In September 1985, the High Sierra Group (including DEC, Hitachi, Phillips, 3M, Apple, AT&T, Xebec, and Microsoft) agreed upon logical file and directory structures for CDROM.[1]

Emerging CDROM Applications

The promise of CDROM lies in electronic publishing and database distribution. CDROM capacity of 550 Mbytes equals 150,000 pages or 250 large books. Books and CDROM are both mass-produced read-only media. This likeness is central to the design of CDROM applications.

The cost of each CDROM edition is $3000 for mastering, with disks costing between $5 and $25 depending upon the size of the edition. Turnaround time is constantly dropping and is quickly approaching the limit with estimates as low as one day.

The economics of publishing, the emergence of standards, and large storage capacity establish the tremendous potential of CDROM for electronic publishing and database distribution. This potential is heightened by the large installed base of PCs.

Many corporations maintain centralized databases (e.g., parts catalogs, inventory data, on-line medical services) that are accessed by geographically distributed users. The cost in time and money to access these databases using telephone lines can be quite high, and query bandwidths are limited by the database service bandwidth. Mailing CDROM copies of a database to users provides an attractive alternative if the database contents do not change rapidly. This migration from on-line to on-disk information retrieval is fostered by the compact size and improved publication turnaround available with CDROM.

CDROM is also well suited for consumer and proprietary software distribution. CDROM offers manufacturing advantages over magnetic tape and floppy disk. Copies can be mass-produced instead of individually written byte by byte. CDROM transfer rates are faster than floppy disk and comparable to many tape products.

[1] The High Sierra Ad Hoc Advisory Committee issued the working paper for a standard CDROM volume and file structure on May 28, 1986. This standard is currently under review by the European Computer Manufacturer's Association (ECMA) and the National Information Standards Organization (NISO).

Write-Once Optical Memory

While CDROM is the optical disk analog for books, WORM disk functionality more closely resembles that of paper. The write-once characteristic is a unique attribute that dominates application considerations. Consequently, developers should consider applications where the media indelibility is either a virtue or an insignificant factor.

Data is written on WORM media by a laser in high-power mode, ablating, or melting, a hole in the reflective recording layer. A thin film of tellurium oxide serves as the reflective recording surface. Figure 4 illustrates WORM writing and reading. The write data modulates the writing laser output. Laser diodes that support MHz modulation frequencies are required for WORM drives.

The data is read in the same way as CDROM with the laser in low-power mode. The reduced power of the laser results in nondestructive readout and supports repeated reads without degrading data integrity. Typically, WORM drives use a single laser for both read and write. Laser power is typically 1 milliwatt for read and 8 milliwatts for write.

Because the media are write once and cannot be factory tested for writeability of each sector, WORM drives are designed to verify written data and rewrite data to spare sectors when errors are detected. One or two sectors per track are typically allocated for sparing, and additional spare tracks are available in case track-resident spare sectors are exhausted.

There are two methods of write verification. Direct read during write (DRDW) verifies the data as it is written, while direct read after write (DRAW) checks the written data on the next revolution. DRAW write transfer rates are no more than half the read rate, while DRDW read and write rates are equal.

Figure 4

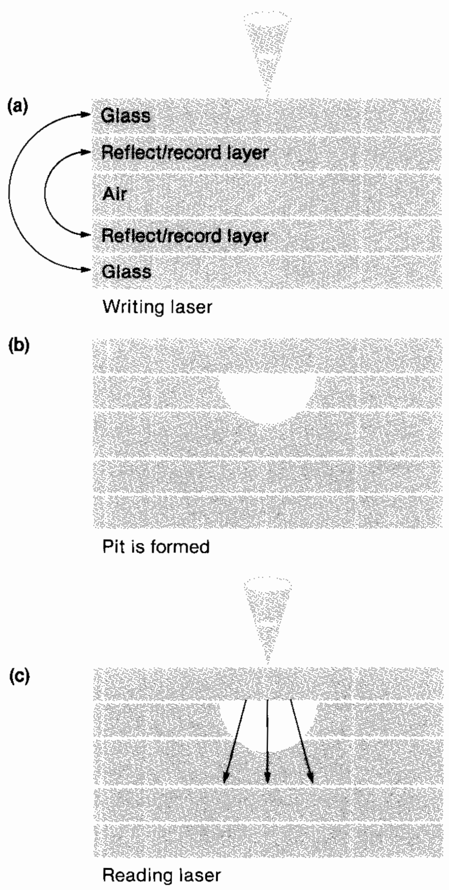


Figure 4.

WORM write and read. (a) The writing laser activates the recording layer. (b) A pit is formed. (c) The reading laser sees anti-reflection and interprets as data. Note that the media shown are two-sided.

The write-once characteristic complicates software integration. CDROM can be viewed simply as write-protected disks with directories complying with the High Sierra standard. WORM directory entries cannot be overwritten to support updates and WORM resident directories are often organized as directory audit trails. Fast access to files on WORM requires magnetic disk or RAM caching of WORM directories, especially when a large number of files reside on a disk.

WORM Product Lines

Several WORM systems are available or in development. The 12-inch product currently has drives and media in volume production. The 5¼-inch product is emerging with limited availability of drives for evaluation. No vendors have announced a 3½-inch product, but media and drives in this attractive size are inevitable. Each of these products has unique performance characteristics, costs, and standards status.

12-inch WORMs. Two to 3 gigabytes can be stored on a double-sided 12-inch disk. Read transfer rates are between 250 and 400 Kbytes per second; write transfer rates depend on the selected verification technique. Worst-case seek times are 250 milliseconds. Seeks to adjacent tracks, supported by the fine position actuator, take less than a millisecond.

Drives with SCSI and the General Purpose Interface Bus (GP-IB or IEEE 488) interfaces are available in 19-inch rack-mountable form factors. Media cost per megabyte is between 10 and 15 cents. No standards have evolved, and the technology is too firmly entrenched for standards to emerge at this late date.

5¼-inch WORMs. Agreement on standards is a gating factor for the emerging 5¼-inch WORM product line. ANSI technical committee X3B11 is working on standards that support media interchangeability.

Several vendors are taking the risk of introducing 5¼-inch WORM products before agreement is reached. Media interchangeability among some Japanese offerings is the result of adoption of two standards by the Japan committee. Capacities of 600 Mbytes and 800 Mbytes are available on double-sided media. Access times are in the 100-millisecond range. Vendors are offering drives with SCSI and IBM PC interfaces. Media costs are around 10 cents per megabyte.

WORM Jukeboxes for On-line Archives

While WORM capacities make the technology attractive in terms of storage cost per bit, the ability to remove media from drives facilitates the organization of drives, cartridges, and interchange mechanisms that are not feasible with magnetic disks. The "jukebox" organization of drives, media storage, and changers defines a new storage capability. (See Figure 5.)

In jukeboxes, the robotic pickers automate cartridge access and reduce access times by 2 to 4 orders of magnitude when compared to tape archives. Jukebox change times currently range from 10 to 20 seconds. These access times are closer to traditional on-line access times (milliseconds) than off-line access times (hours). Jukeboxes bring archives on-line and facilitate much larger, more cost-effective online databases than previous technologies.

Magnetic disk average access times are the sum of average seek time and average latency. Latency is 8.3 milliseconds, and seek time is 15 milliseconds for state-of-the-art, 800-Mbyte, 9-inch Winchester disks. When designing database systems, access time, megabytes per spindle, and queuing strategy determine the number of transactions that can be completed per unit time. Seek optimization increases service capacity.

For jukeboxes, access time is the sum of mount time, seek time, and latency. Mount time dominates access time. Measured in seconds, it includes demount when all drives are loaded, and becomes zero when the desired volume is resident in a drive.

Jukebox vendors stress change time (time to unload and load a cartridge to a drive) as the figure of merit along with capacity. System managers must balance gigabytes per cartridge and drives and cartridges per changer to support service goals. Mount optimization must be part of advanced queuing strategies used to optimize jukebox service bandwidth.

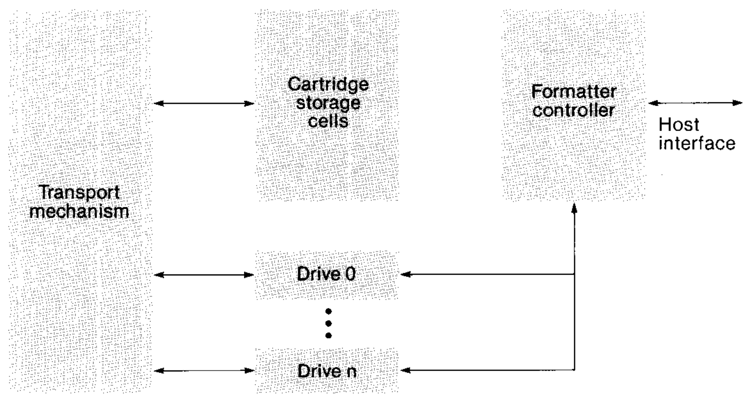
Jukebox Applications

The first jukebox applications were standalone electronic file cabinets. The very low storage costs justified the use of WORM media for paper replacement. This product addresses the high-priority business need to control paper and reduce storage space requirements.

There are many computer application opportunities for jukeboxes. The migration of tape archives to jukeboxes greatly reduces data management costs while improving data accessibility. Not only are handling costs for volume mount and demount reduced, but the ten-year WORM lifetime eliminates the need to rewind media every two years as required by tape.

Media accessibility increases the value of information. For example, the costs to analyze and correlate very large pharmaceutical test-result databases are prohibitive today but will become economically feasible when the data is stored on jukeboxes.

Figure 5



Much information that is archived on tape today by Tandem customers is of a write-once nature. Financial and transaction audit trails and medical, legal, shipping, tax, and insurance records do not change over time, although information is often appended. Files stored in jukebox archives can be accessed quickly to enhance service. The indelibility and media lifetime of WORM storage is attractive for retaining data for long intervals.

Figure 5.

The jukebox organization of drives, media storage, and changers. Jukeboxes automate media handling, thus defining a new storage capability.

Jukebox storage costs are low enough to justify the migration of paper, microfiche, and microfilm databases to WORM media. Data stored in a jukebox is less likely to be misplaced. More importantly, the data resides in a computer environment and can be processed and accessed in a distributed computer network. This is more powerful than document storage and retrieval provided by electronic file cabinets. Systems with these features are of great value for engineering and manufacturing documentation where current solutions are sought for tracking updates and managing huge volumes of data.

Tandem's 5200 Optical Storage Facility (OSF)
The Tandem 5200 OSF, 3210 companion controller, and optical disk system software were announced in October 1987. The products became available in December 1987. This product provides great utility to Tandem systems with large tape archives by bringing the archived data on line and reducing operator costs.

Providing 84 gigabytes of on-line WORM storage, the 5200 OSF contains up to 32 disks, two WORM drives, and an automatic disk changer. The changer can unload and store one cartridge and select and load another cartridge in under 17 seconds.

The 5200 OSF is connected to the 3210 controller by a differential SCSI interface; it can be located up to 75 feet from its host. The 3210 controller is able to read and write data at the maximum device rates: 300 Kbytes per second for read and 150 Kbytes per second for write using DRAW. The controller uses redundant lockstepped 68000 microprocessors with proprietary VLSI checking logic to enhance data integrity.

The Tandem OSF-DPO (Disk Process Optical) software is available with the C00 release of GUARDIAN and supports the same programmatic interface as Tandem magnetic disk software. This allows application programs to use either optical or magnetic disks without program modification. The initial release supports file copy between optical and magnetic disk. In addition, unstructured read/write access to optical disk is provided.

Optical Storage and the Storage Hierarchy

The traditional storage hierarchy, a reality dictated by cost, performance, and reliability, is shown in Figure 6. The cost of on-line storage has traditionally forced archives off-line. Figure 7, showing the modern storage hierarchy, illustrates the impact of optical disk technology. The line between on-line storage and archive is blurred as WORM jukeboxes make on-line archives cost-effective.

Although optical disk is displacing magnetic tape for archiving functions, it cannot replace tape for all applications. BACKUP and RESTORE require high data-transfer rates because backup windows are shrinking. Media reuseability is desirable when backups are done at frequent intervals and there is no need to archive obsolete versions. State-of-the-art tape products transfer read data ten times faster than WORMs. This performance difference and the write-once nature of WORMs generally prevent optical disk from being a cost-effective choice for backup and restoration of computer data. Tape will continue in this role. ANSI tape standards also make tape a universal medium for interchange of computer data, but the emergence of WORM interchangeability standards will affect this tape function.

Optical disk is also suitable for storage functions previously filled exclusively by paper. CDROM is a viable publishing medium for professionals and for corporations already employing PCs. Because of the significant entry cost of optical storage, not all paper applications will migrate to the new media.

Magnetic disks are the only solution for high-speed on-line transaction databases. The information stored in these databases is very dynamic, and the overwrite capability is essential. The high-performance characteristic of magnetic disk is also required to support query bandwidths. Low-end magnetic disks are vulnerable to optical disks. When introduced, erasable optical disk could erode the floppy disk market and increase the power of PCs.

Conclusion

No current storage medium is eliminated by optical storage. Optical disk coexists with current storage technologies, and much of the market for optical disk is incremental. By decreasing on-line storage costs and bringing large amounts of data into the computer environment, optical disk technology increases the usability and value of information.

Reference

Ng, D. 1986. Plated Media Technology Used in the XL8 Storage Facility. *Tandem Systems Review*. Vol. 2, No. 2. Tandem Computers Incorporated. Part no. 83937.

Acknowledgments

I want to thank Steve Coleman, Bruce Lowenthal, and Anand Patel for their valuable input and comments.

Lauryl Sabaroff is manager of Future Storage Products, Hardware Development. She holds a BSE with high honors from the University of California at Irvine. Before joining Tandem she was a consulting engineer designing telecommunications processors. She also worked for a major mainframe manufacturer in I/O controllers.

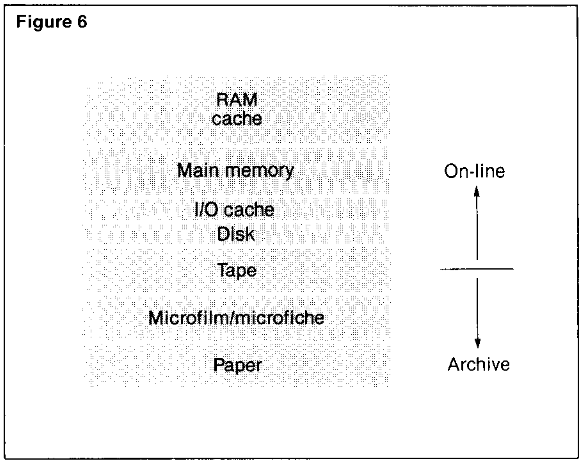


Figure 6.

Traditional storage hierarchy. Archives are traditionally off-line, a reality dictated by the economics of cost, performance, and reliability.

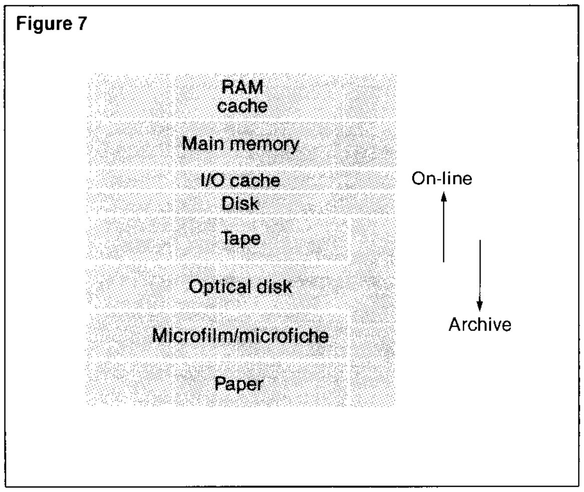


Figure 7.

Modern storage hierarchy. Optical storage brings archives on-line.

Technical Paper: Tandem's Approach to Fault Tolerance

Systems that never fail are becoming essential in all sectors of the economy and are affecting all aspects of society. Fault-tolerant systems are required for life-critical applications such as hospital patient-monitoring systems and for what James Martin has termed "mission-critical" systems, where the computer system is essential to the success of the business. An example of a mission-critical system is the New York Stock Exchange trading system: if the system fails, the exchange must close.

If these were the only categories of systems that required high availability, fault-tolerant systems would remain relegated to a rather small market niche. However, we live in an age where information itself provides the competitive advantage for businesses and often is, in fact, the actual asset of the business. As Daniel Bell states, "In an industrialized society, the strategic resource is capital. In our new society, the strategic resource is information."

The information age is characterized by the elimination of the middleman. New applications of information processing that provide goods and services directly to the consumer with few or no intermediaries include automated teller machines, immediate point-of-sale debiting of bank accounts, home banking, and home shopping. Highly available, reliable transaction processing systems are a fundamental component of all of these applications. Most people are willing to use such services only because the current generation of transaction processing systems provides acceptable levels of responsiveness and dependability. In a sense, the service should be viewed as an electric power utility: while the presence of its output is taken for granted, its absence is painfully obvious.

Tandem's NonStop Architecture

John Von Neumann is credited with the observation that a very reliable system can be built from unreliable components. Von Neumann showed that by using redundant components, thus eliminating single points of failure, the mean time between failure (MTBF) of a system as a whole can be many times the individual MTBFs of its components. A typical disk drive, for example, fails about once a year (10,000 hours). If, however, two disks are treated as a single mirrored or duplexed logical volume, both disks in the pair will be down at the same time about once every 5,000 years (assuming a mean time to repair [MTTR] of 10 hours).

Tandem systems ensure both high availability and high performance because each system is configured with parallel hardware components and software processes. The key to the architecture is that there is no single hardware component or software module whose failure can bring down the system. The multiplicity of paths, components, and processes makes it possible for the system to continue to operate despite the failure of an individual module. There is always an alternate hardware component or software process that will take over the module's function.

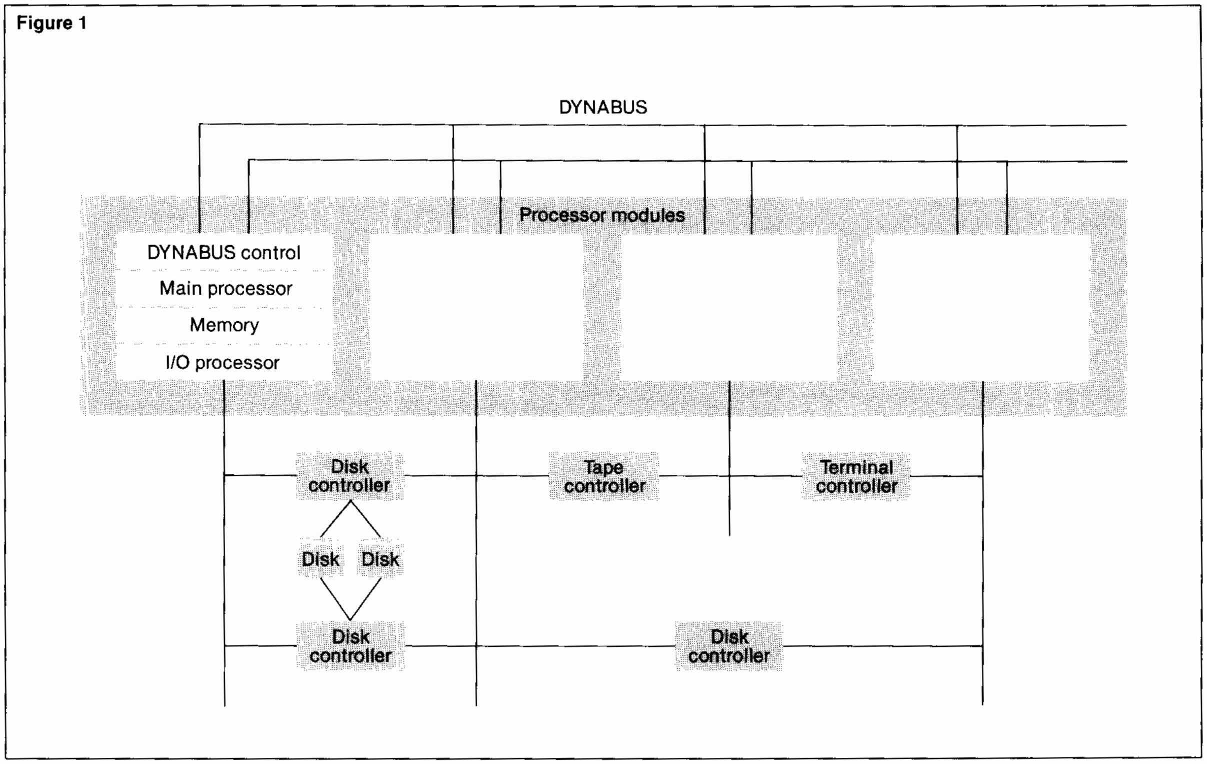
Hardware Components

The Tandem system is designed to be efficient and cost-effective as well as fault-tolerant. To achieve this result, it uses its major parallel components independently and concurrently rather than in lockstep or as idle or "hot" standby backup components that are used only for the duration of a failure. This strategy optimizes both performance and cost/performance. While all modules are functioning, the system provides much better performance than it would given the same set of components being used in a lockstep or hot standby configuration; during a failure situation, the system still performs as well as it ever would have under either of the other two approaches. Parallel components handle their workloads much like riders on a two-seater bicycle: both riders pedal, contributing to the speed of the vehicle, but either one can keep the bicycle moving if the other should pause.

The dual-channel DYNABUS, dual-ported controllers, and mirrored disks all are examples of the implementation of this strategy. As shown in Figure 1, each processor module is connected to all other processor modules in its system by the DYNABUS. The DYNABUS is a pair of high-speed interprocessor buses. Each bus is fully autonomous, operating independently of and simultaneously with the other bus in the pair. The use of two buses ensures that two data paths exist between all processor modules in the system. Normally, both buses are used for communication between processors. If one bus fails, the GUARDIAN 90 operating system automatically reroutes all interprocessor communication over the remaining bus. Because each bus is controlled by its own bus controller and each controller is independent of the logic circuits within the processors, a processor failure cannot stop bus transmission.

A processor transfers data to an I/O device such as a disk drive over an I/O channel (I/O bus). There is one I/O channel per processor. Individual devices are connected to the channel via device controllers. Each controller has two separate ports, allowing it to be connected to two processors' I/O channels. At any given time, one of the two processors manages the controller; if the managing processor fails, the other processor immediately takes ownership of it. Some devices, such as disk drives, can be connected to two separate controllers.

Figure 1.
NonStop hardware architecture.



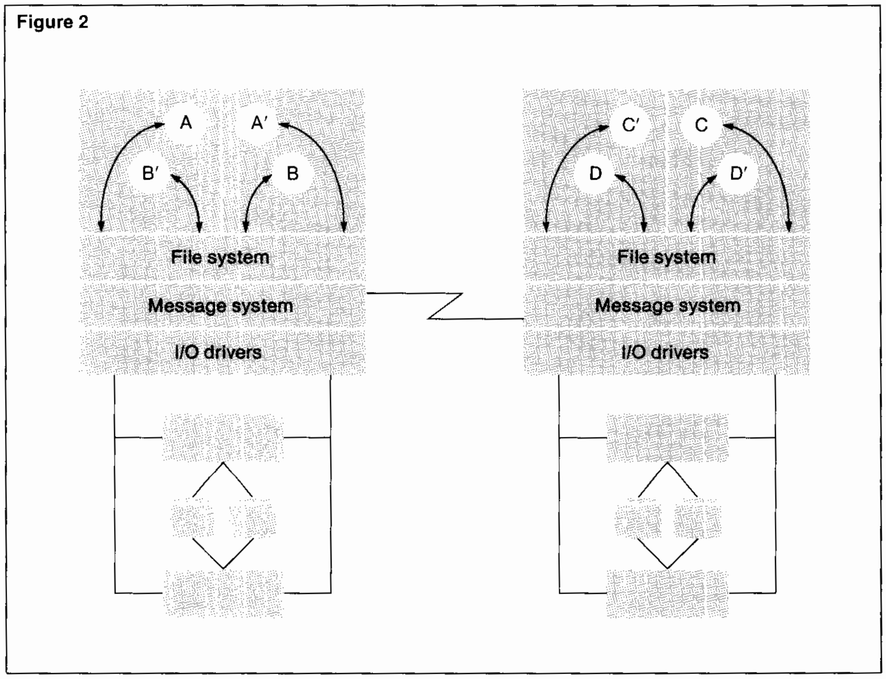


Figure 2.
NonStop software architecture.

The many possible combinations of ports and I/O paths allow a system's disk drives and controllers to be configured in ways that ensure an extremely high level of availability. For example, disks can be configured as "mirrored" volumes. As illustrated in Figure 2, a mirrored volume is a pair of physically independent disk drives, usually attached to two separate controllers, that is accessed as a single device and managed by a single I/O program. All data that an application directs to one drive is written to both drives. Since both drives contain identical information, all data that an application reads from a file may be read from either drive. This improves performance, since separate reads may be in progress simultaneously on the two halves of the mirrored pair.

If one of the controllers or drives fails, the other remains in service and processing continues without interruption. If a drive fails and is repaired, its contents are updated online once it is reintegrated into the system, again without interruption to the application.

These are only a few examples of the use of parallel but independent hardware components. The Tandem system includes many other such sets of components covering other vital areas (e.g., power distribution).

Software Structure

The GUARDIAN 90 operating system is designed as a set of separate processes that communicate via messages. This abstraction makes it possible for two processes to interact without caring where they are located with respect to each other—in the same processor, the same system, or the same EXPAND network.

A copy of the GUARDIAN 90 operating system resides in each processor module. This feature contributes to continuous system availability by allowing each processor to act as an individual computer that functions autonomously from other processors within the system. GUARDIAN 90 in turn manages communication between processors over the DYNABUS to provide continuous availability at both the system and application levels.

Process-pairs and checkpointing are the next higher level of underpinnings for continuous availability. The primary process within a process-pair performs the application function. While doing so, it periodically sends checkpoint messages containing its state changes over the DYNABUS to a backup copy of itself that is running in a different processor. The backup process in turn incorporates the state changes into its own environment. This relationship is illustrated in Figure 2.

If the primary processor or process fails due to component failure or software error, the backup process takes over as the new primary and resumes work at the point of the last checkpoint received from its brother process. The new primary process continues to perform the application function throughout failure and repair.

State checkpointing is an effective technique for implementing software-level fault tolerance. Because a backup process is passive, its demands on the resources of the processor where it resides are relatively light.

The availability considerations that apply to the hardware components also apply to process-pairs. Having a backup process take over for a failing primary process gives a mean time to recovery that can be measured in seconds, providing very high availability for the application or system function.

Detection of Processor Failures

The GUARDIAN 90 message system in each processor is responsible for keeping every other processor in the system informed that it is operating correctly. It carries out this function by periodically sending a brief "I'm alive" message over the DYNABUS to the other processors. Each processor in turn periodically checks for receipt of an "I'm alive" message from every other processor. If a processor fails to send its status message, the remaining processors declare that processor "down" and refuse to process further messages from it until it has been reloaded and reintegrated into the system. This prevents a failing processor from contaminating the rest of the system and allows the surviving processors to determine that they must take corrective measures such as assuming ownership of the failed processor's controllers and notifying the backup processes of programs that were running in the failed processor that they should take over.

Reliability

The use of paired hardware or software components improves availability. However, mission-critical systems require high reliability as well as high availability. A distinction must be made between a module failing and the module continuing to work but in an erroneous fashion. A module that fails does not respond at all. A module that malfunctions does respond—but its response may be incorrect and therefore unreliable. It is important that an on-line transaction processing system produce reliable results even in the presence of a failure. This is accomplished by rapid error detection and by damage containment.

Tandem systems are designed for both high availability and high reliability, assuring the user that if the system is operating, it is operating correctly. The system provides reliability through a combination of rapid error detection (self-checking, "fail-fast" modules) and error containment. If a module encounters an error, it either isolates itself or is isolated by another part of the system so it will not corrupt any other module.

The system's structure provides a high degree of insulation (and hence error containment) between software modules. A process does not share any state with other processes; instead, it communicates through messages carried on its behalf by GUARDIAN 90. Thus, each system or application process runs independently. Hardware modules are also independent of each other and do not share critical states with other components. As examples, processors do not share memory with one another; all critical components have backup power supplies; and each device controller has two ports so if one I/O channel fails, the other port can be used to transfer information over the other I/O channel.

System processes and critical hardware modules operating under GUARDIAN 90 are fail-fast; that is, if they do not perform to specified standards, they halt. Both the hardware and software are made fail-fast through error detection. Hardware components use techniques such as parity checking and periodic self-testing. GUARDIAN 90 performs rigorous internal consistency checks to verify its inputs, outputs, and data structures. In the extremely rare instance where an error occurs within a system process or GUARDIAN 90 detects a corrupted data structure, it halts that processor and lets the backup processes in other processors take over. This ensures that no malfunctioning system process continues running once it has detected an error.

The system's architecture also prevents an errant application process from corrupting any data outside of its environment. When such a process calls itself to the system's attention by attempting to perform an illegal operation, GUARDIAN 90 aborts the process and allows the other processes in its processor to continue to run.

Hardware Errors

GUARDIAN 90 checks for hardware malfunctioning by sending out periodic status probes that check the health of each module. The hardware modules themselves perform extensive self-tests both as part of normal operation and when they are otherwise idle. When an error occurs, the module either reports it to GUARDIAN 90 for resolution or takes itself out of service.

In some instances, critical modules are able to correct errors and proceed rather than halt. For example, if an error occurs in main memory, the processor detects and corrects the error using an error correcting code (ECC). If the error is too severe to correct, GUARDIAN 90 must decide what action to take, as in the following examples:

- When a word of main memory gets a single-bit (correctable) error, the processor detects it. It then uses the ECC information to derive the correct data and rewrites the word.
- When a word of main memory gets a double-bit (uncorrectable) error, the processor generates a hardware trap and lets GUARDIAN 90 determine the response. If the word is contained in a page that is private to an application process, the system software checks whether an up-to-date copy of that page is available on disk. If so, GUARDIAN 90 rereads the page from disk into a different page of memory and allows the process to resume execution; otherwise, it traps out the process. The usual result is that the process aborts execution, letting its backup take over.

If the word is contained in a page containing critical system data, then the consequences of not having an up-to-date disk copy of the page are more severe: GUARDIAN 90 logs the error and halts the processor. As a result, the remaining processors notice the absence of "I'm alive" messages, declare the processor to be down, and take appropriate corrective action.

Since memory is not shared between processors, the memory contents of the other processors are not affected by the error.

Software Errors

Even if it has been carefully constructed and tested, a complex module of software may still contain a few errors (bugs). These errors may be classified into two types: hard errors and transient errors. Hard errors recur each time the module is run. Because they are reproducible, they usually are relatively easy to find. Transient errors, on the other hand, are difficult to find because they are timing-related and do not occur each time the module is run; they only occur when a certain set of conditions exists at a particular point during execution.

Most hard errors are unearthed during testing or within a short period of production use; errors that appear after a module has been in production for a long time are more likely to be transient than hard. Out-of-resource errors are an exception, since they may lie dormant until triggered by a growing workload.

As is the case for other mainframe computers, Tandem's system programs contain millions of lines of code. It is not feasible to test all possible permutations and combinations of system program execution and configuration. The inevitable result is that the code still contains a few transient errors even after months of systematic testing, debugging, and use.

However, unlike other mainframe computers, Tandem systems have a high degree of built-in tolerance to transient software errors. The design principle for preventing the system as a whole from halting is that each processor operates as a separate system with its own mix of executing programs, internal timing, memory, and I/O channel. Should a system software error cause a process to fail, its backup process in another processor module takes over and begins to execute at the most recent checkpoint. Because the environment of the backup process is different from that of the primary process, and transient errors are triggered by specific sets of conditions, the backup process is unlikely to encounter the same bug. This form of fault tolerance is not available in either single-CPU systems or multiprocessor systems whose processors all execute the same instruction stream in lockstep.

Reduction of Application Complexity

It is easier and cheaper to avoid introducing errors than to uncover, isolate, and fix them. One way to improve an application program's reliability and availability is to reduce its complexity.

Since developing and debugging an application that runs correctly as a process-pair is slightly more difficult than doing so for its non-fault-tolerant equivalent, there are benefits in automating support for fault tolerance. An application developer should be able to focus on the business unit of work to be performed without also having to determine when and what to checkpoint. This has been the case since 1981, when Tandem introduced the Transaction Monitoring Facility (TMF). Development of applications under TMF reduces complexity in two ways: it provides fault tolerance without requiring the application programmer to handle checkpointing, and it supports the construct of a transaction.

A transaction is defined as a transformation of the database from one state to a new state with the properties of atomicity (either all changes take effect or none take effect), durability (the effect survives failures), consistency (a correct transformation is performed), and isolation (locking is used to ensure that the transaction is unaffected by concurrent transactions).

To take advantage of TMF, all that the application programmer must do is to determine what constitutes a transaction and then bracket the code that carries it out between a BEGIN-TRANSACTION verb and an END-TRANSACTION verb. The programmer can respond to abnormal conditions that may arise during the transaction by executing an ABORT-TRANSACTION verb, in which case TMF will remove all effects of the aborted transaction.

TMF satisfies the requirements of transactions by using a locking scheme and a write-ahead protocol. Audit trail records are recorded on disk before the END-TRANSACTION is allowed to successfully complete and the application is allowed to proceed. Changes to the database may be written asynchronously from the transaction and in fact may occur at a much later time. TMF itself uses process-pairs to recover from processor or system failures. When a transaction updates data on multiple nodes, TMF uses a two-phase commit protocol to ensure consistency among all nodes.

TMF has an additional benefit: its use not only simplifies application design but also extends fault tolerance to protect against multiple failures. As an example, if both the primary and mirror disk volumes on which a database resides suffer simultaneous head crashes, TMF is able to recover the data. First, it restores a previous image of the database from an on-line dump copy. It then uses the audit trail contents to reapply all database changes that took place between the time of the on-line dump and the time of the failure. Table 1 summarizes the functions that TMF provides and the ways in which it extends the fault tolerance of Tandem systems.

Table 1.
TMF recovery mechanisms.

Types of failures	TMF mechanism to recover
Application	Transaction backout
Hardware Power System software	Autorollback
Both a disk volume and its "mirror" fail	Rollforward

Why Do Systems (Still) Fail?

It is vital that designers of systems that are meant to be highly available and reliable understand what causes system failures. In 1985, Dr. Jim Gray of Tandem Computers performed a study that examined system outage data from more than 2000 systems representing over 10,000,000 system hours. Administrative and software errors predominated; hardware and environmental errors were minor contributors to system outages. The results of this study are summarized in Table 2.

Because the study was based on outages reported to Tandem by customers, it is likely that administrative, application-caused, and environment-related failures are under-reported. A more thorough study that is now in progress suggests that environment-related failures (e.g., floods and lightning strikes) go unreported 50% of the time.

Table 2.
A summary of system failure.

System failure mode	Probabilities	MTBF (years)
Administration	42%	31
Maintenance	25%	
Operation	9%	
Configuration	8%	
Software	25%	50
Application	4%	
Vendor	21%	
Hardware	18%	73
Environment	14%	87

Even though the original study covers only a subset of all outages, there are some valid conclusions that can be drawn from it. The principal observation is that hardware is becoming increasingly reliable, while software is becoming more complex and responsible for a larger portion of both system function and the organization's business. As this trend continues, the impact of administrative errors also increases.

This leads to the inescapable conclusions that fault-tolerant solutions that address hardware failures alone are inadequate, and that requirements for improving systems availability include reduction of administrative errors as well as provision of hardware and software fault tolerance.

High System Availability

Self-Help

A system manager can take many measures to reduce the likelihood of the system crashing due to operational or administrative error. Options include writing operations guides, providing formal training for operators, automating command sequences by setting up TACL (Tandem Advanced Command Language) routines or OBEY files, and by performing disaster recovery drills. These options have a common denominator: administrative errors are minimized when a site has instituted a well-thought-out set of procedures to handle both normal operations and error conditions and has implemented them in a manner that makes them easy to follow even in the face of disaster.

A site can also reduce the need to schedule outages for system reconfiguration by planning ahead for capacity growth and preconfiguring additional resources accordingly. Tandem's Dynamic System Configuration tools, which became available in 1987, lessen the need to preconfigure software changes by allowing new devices to be added on-line.

The best way to avert unplanned outages is to detect and react to potential problems before they grow into full-scale crises. Problem detection and resolution are sometimes regarded as an art form rather than a science, but this situation is changing.

Diagnosis and Repair of Hardware Errors

In the study performed by Gray, maintenance errors accounted for the single largest source of system failures. The Tandem Maintenance and Diagnostic System (TMDS), which was introduced in 1984, begins to address this category of problem.

TMDS improves availability in several important ways. First, it allows for on-line diagnosis and repair of failed components while other components continue to provide service, thereby reducing the need for system downtime. Scheduled outages often are not included in failure statistics, but in environments where absolutely no loss of service is acceptable or where it is vital that loss of service be minimized, even a scheduled outage for maintenance must be counted as a failure.

TMDS introduced the use of expert systems technology to analyze error information and identify the failing field-replaceable unit (FRU). Tandem hardware and low-level system software generate messages, known as event signatures, when any unusual event occurs. Some examples of events that trigger signature generation are the temperature within a processor cabinet rising above the operating range, a disk drive having to retry an operation, or the processor detecting a correctable or uncorrectable memory error. The resulting event signatures are automatically recorded in an event log for analysis by the expert-system software.

TMDS is able to successfully identify the failing FRU over 90% of the time. When a TMDS fault analyzer detects a failure, it has the ability to sound an audible alarm and optionally automatically dial out to a remote service center. This can significantly decrease the MTTR by providing rapid notification and improving the likelihood that the CE will arrive with the right set of replacement FRUs in hand. Reducing the MTTR also lowers the probability of a second failure occurring before the first failed component can be repaired and placed back in service.

The TMDS event analysis system allows it to monitor system hardware for signs of impending failures. For example, if a device's retryable error rate indicates that its performance is degrading, TMDS can initiate diagnostics against the device, isolate the problem, and determine whether preventive maintenance or FRU replacement is necessary. This capability allows service to be scheduled and carried out at a relatively convenient time and helps to reduce the duration of a service outage.

TMDS also improves availability by guiding the CE through the repair and reintegration process. This feature significantly reduces the chance that the CE will inadvertently trigger a second error, possibly causing a complete system failure, while the system already is in a vulnerable state.

Minimizing Errors

As is the case for hardware repair and reintegration, operational miscues that take place while responding to a system problem can introduce further problems or even cause a complete system failure. This kind of situation also may arise if an operator makes a mistake while carrying out a procedure when the system is functioning normally. Ways that the system can reduce risk in this area include automating operational procedures insofar as possible and providing guidance in resolving problems that cannot be handled without operator intervention.

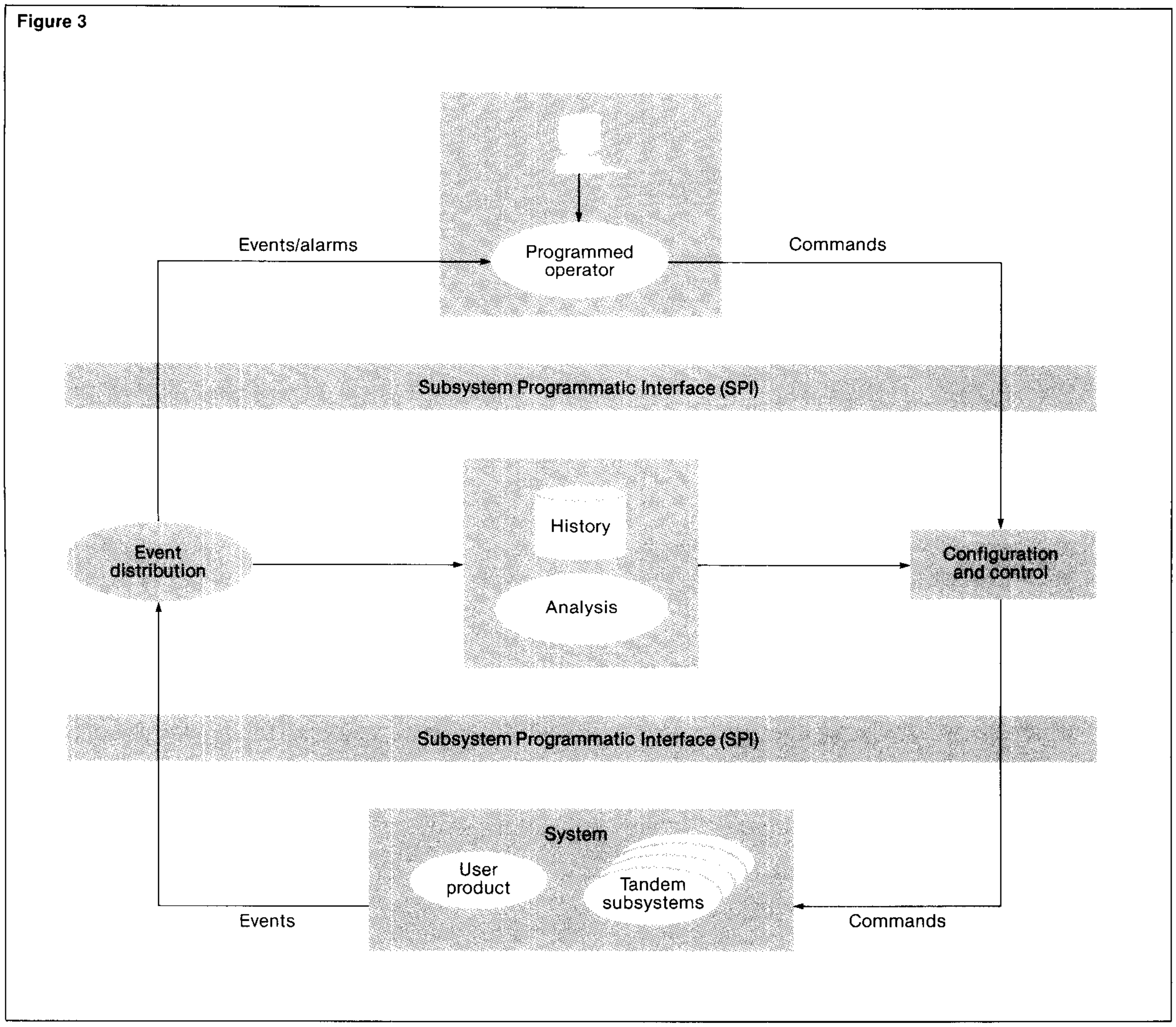


Figure 3.
Distributed Systems Management.

Tandem's Distributed Systems Management (DSM) architecture, introduced in 1987, supplies both improved reporting of system events and tools that allow automated response to events. The flow of information through DSM's components is illustrated in Figure 3. All major subsystems have been enhanced to take advantage of two new facilities, the Subsystem Programmatic Interface (SPI) and the Event Management System (EMS). The EMS facilities allow a network/system management application to receive notification of changes in system state. The inclusion of SPI support in Tandem subsystems allows the application to respond to a change of state by interacting programmatically with the appropriate subsystem. For example, a DSM application might react to notification of a line going down by using SPI to ask the appropriate communications subsystem to restart it.

Figure 4

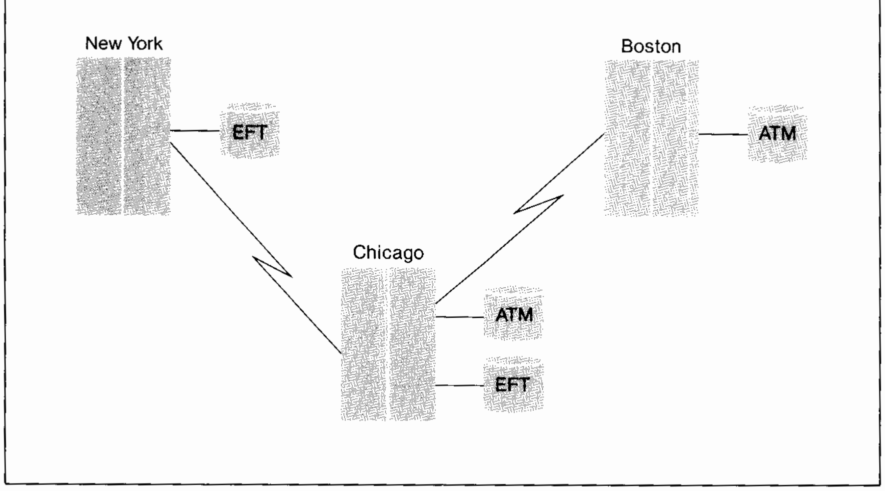


Figure 4.

Remote Duplicate Database Facility (RDF). The system in Chicago is used to provide backup for applications running on systems in New York and Boston.

This approach can be extended to support remote, unattended nodes. Under DSM, either all or a filtered subset of a system's events can be forwarded to a DSM application running on a different system, allowing multiple systems to be managed from a single location.

DSM applications can greatly reduce system operation complexity, which in turn greatly reduces the risk of human error.

High Application Ability

For some applications, minimizing a system's downtime is not adequate; the business also must be protected against a total site failure. Tandem added support for site-level or geographic fault tolerance with the introduction of the Remote Duplicate Database Facility (RDF) in 1987. RDF is an on-line database utility that can help speed the process of getting an application up and running at a backup site when its primary site suffers a complete disaster such as a fire, flood, or earthquake. Figure 4 illustrates a system in Chicago being used to provide disaster backup for applications running on systems located in New York and Boston.

RDF functions as an extension of TMF by sending the afterimage audit records of completed transactions over the network to a remote backup system. The audit records are applied to the backup system's copy of the application database. When disaster strikes, the backup system already has the information necessary to ensure that it can take over application processing against an up-to-date copy of the database.

Toward Continuous Availability

A system can be said to provide truly continuous availability when both the system and the applications that it supports are available 24 hours per day, 7 days per week, 365 days a year. Currently, Tandem systems go a long way toward meeting this goal, but several complex problems remain to be solved before the goal can be attained. This is a daunting challenge, but Tandem's architecture provides a firm base for the extensions that must be developed. Improved capabilities needed to support continuous availability include the ability to replace both system- and application-level software on-line, the ability to expand an existing system on-line without extensive planning, and a fully developed set of tools for automatic operations.

Conclusion

Tandem's fault-tolerant architecture is based on a set of related constructs implemented at various levels within the system's hardware and software. At the lowest levels, parallel but independent components ensure that the loss of a single component will not halt the system. Since a mission-critical system must be both available and reliable, hardware and software modules are designed to detect, report, and contain errors in a fail-fast manner. Process-pairs and checkpointing provide a means of recovering from processor failures and transient software errors. Higher-level software such as TMDS and DSM provides tools for recognizing potential hardware and software problems and resolving them before they become critical. Finally, TMF obviates the need for application programmers to concern themselves with process-pairs and checkpointing and also serves as the foundation for RDF's geographic fault tolerance.

Tandem has developed a combined hardware and software architecture that not only provides high availability but has successfully evolved to meet the ever-increasing demands for continuous availability.

References

Allen, J. and Boyle, R. 1987. The VLX: A Design for Serviceability. *Tandem Systems Review*. Vol. 3, No. 1. Tandem Computers Incorporated. Part no. 83939.

Bartlett, J. 1981. A NonStop Kernel. Proceedings of the Eighth Symposium on Operating System Principles, pp. 22-29, December. Also Tandem Computers TR81. 4.

Bartlett, J., Gray, J., and Horst, B. 1986. Fault Tolerance in Tandem Computer Systems. Tandem Technical Report TR86. 2. Tandem Computers Incorporated.

Bell, D. 1976. *The Coming of the Post Industrial Society; A Venture in Social Forecasting*. Basic.

Blain, C., White, L., and Witte, W. 1987. Enhancements to TMDS. *Tandem Systems Review*. Vol. 3, No. 2. Tandem Computers Incorporated. Part no. 83940.

Borr, A. 1981. Transaction Monitoring in ENCOMPASS. Proceedings of the Seventh International Conference on Very Large Databases, September. IEEE Press. Also Tandem Computers TR81. 2.

Brooks, R. 1985. An Approach to High Availability in High-Transaction-Rate Systems. *IBM Systems Journal*, Vol. 24, Nos. 3/4, pp. 279-293. International Business Machines Corporation.

Gray, J. 1981. The Transaction Concept: Virtues and Limitations. Proceedings of the Seventh International Conference on Very Large Databases, September. IEEE Press. Also Tandem Computers TR81. 3.

Gray, J. 1985. Why Do Computers Stop and What Can We Do About It? Tandem Technical Report TR85. 7. Tandem Computers Incorporated.

Troisi, J. 1985. Introducing TMDS, Tandem's New On-line Diagnostic System. *Tandem Systems Review*. Vol. 1, No. 2. Tandem Computers Incorporated. Part no. 83935.

Von Neumann, J. 1956. *Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components*. Automata Studies. Princeton University Press.

Acknowledgment
The authors would like to thank Jim Gray for his careful review of multiple versions of this article.

Wendy Bartlett has been with Tandem since 1978 and has been a member of the Operating Systems Group since 1979. Before joining Tandem, she worked for other vendors as an applications programmer. She has an M.S. in Computer Science from Stanford University.

Brian Ball joined Tandem two years ago and is currently manager of OLTP product management. He has over 14 years of experience in transaction process and operating systems with a major mainframe manufacturer and various software development companies.

TANDEM PUBLICATIONS ORDER FORM

The *Tandem Systems Review* and the Tandem Application Monograph Series are combined in one free subscription. Use this form to subscribe, change a subscription, and order back copies.

For requests *within the U.S.*, send this form to:

Tandem Computers Incorporated
Tandem Systems Review
18922 Forge Drive, LOC 216-05
Cupertino, CA 95014

For requests *outside the U.S.*, send this form to your local Tandem sales office.

Check the appropriate box(es):

□ New subscription (# of copies desired ________)
□ Subscription change (# of copies desired ________)
□ Request for back copies. (Shipment subject to availability.)

Print your current address here:

____
COMPANY NAME

____
ADDRESS

____

____

____

____
ATTENTION

____
PHONE NUMBER (U.S.)

If your address has changed, print the old one here:

____
COMPANY NAME

____
ADDRESS

____

____

____

____
ATTENTION

____
PHONE NUMBER (U.S.)

To order back copies, write the number of copies next to the title(s) below.

NUMBER OF COPIES

Tandem Journal

_______ Part No. 83930, Vol. 1, No. 1, Fall 1983

_______ Part No. 83931, Vol. 2, No. 1, Winter 1984

_______ Part No. 83932, Vol. 2, No. 2, Spring 1984

_______ Part No. 83933, Vol. 2, No. 3, Summer 1984

Tandem Systems Review

_______ Part No. 83937, Vol. 2, No. 2, June 1986

_______ Part No. 83938, Vol. 2, No. 3, December 1986

_______ Part No. 83939, Vol. 3, No. 1, March 1987

_______ Part No. 83940, Vol. 3, No. 2, August 1987

_______ Part No. 11078, Vol. 4, No. 1, February 1988

Tandem Application Monograph Series

_______ Part No. 83900, *Developing TMF-Protected Application Software,* March 1983, AM-005

_______ Part No. 83901, *Designing a Tandem/Word Processor Interface,* March 1983, AM-006

_______ Part No. 83902, *Integrating Corporate Information Systems: The Intelligent-Network Strategy,* March 1983, AM-007

_______ Part No. 83903, *Application Data Base Design in a Tandem Environment,* August 1983

_______ Part No. 83904, *Capacity Planning for Tandem Computer Systems,* October 1984

_______ Part No. 83905, *Sociable Systems: A Look at the Tandem Corporate Network,* May 1985

_______ Part No. 83906, *Transaction Processing on the Tandem NonStop Computer: Requestor/Server Structures,* January 1982, SEDS-001

_______ Part No. 83907, *Designing a Network-Based Transaction-Processing System,* April 1982, SEDS-002

_______ Part No. 83909, *A Multi-Function Network for Business Automation,* May 1982, SEDS-004

TANDEM EMPLOYEES: PLEASE ORDER YOUR COPIES THROUGH YOUR MARKETING LITERATURE COORDINATOR.

TANDEMCOMPUTERS

400103 2/88 Printed in USA

Part No. 11078