THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 6, July–August 1983
Printed in U.S.A.

Modernization of the Suburban ESS:

Adding Data Links to an Existing ESS

By C. E. ISHMAN,* R. B. SANDERSON,* L. M. TAFF,* D. P. TRUAX,* and C. T. TULLOSS*

(Manuscript received June 21, 1982)

As a major modernizing improvement data links for three important functions have been added to the 2B Electronic Switching System (ESS). The data-link system was designed with a layered architecture using an enhanced subset of the X.25 protocol. Levels 1 and 2 of the protocol are implemented in the hardware and software of a new device, the Serial Peripheral Unit Controller/Data Link (SPUC/DL). Within the SPUC/DL itself, check circuitry and software are used for error detection. A second SPUC/DL for each critical application provides redundancy on either a dedicated or dial-up basis. An elaborate diagnostic program within the 2B ESS can be invoked automatically or manually. This paper discusses some of the issues in retrofitting the BX.25 protocol to pre-X.25-speaking machines.

I. INTRODUCTION

This paper describes the addition of real-time data-link capability to the No. 2B Electronic Switching System (ESS)†. The paper combines a technical description of the project with explanations of certain

* Bell Laboratories.

† Acronyms and abbreviations used in this paper are defined at the back of the Journal.

key decisions. We attempt to be candid in assessing both the virtues and pitfalls of such decisions.

The No. 2B ESS is a medium-sized telephone switching system designed for rural and suburban offices with 5000 to 20,000 customer telephone lines. The addition of data links to the system was but one of several major enhancements in the development of a new generic program developed for the 2B—the 2B Extended feature generic #3 (2BE3)—described in accompanying papers.

1.1 Project goals

Our major goal in this project was to create appropriate hardware/software systems to provide data-link capabilities from the 2B to each of three different remote Bell System *applications*. Each application had different characteristics and reliability requirements. An overview of the desired network capabilities is shown in Fig. 1. Note that other types of ESS machines as well as additional 2B ESS offices may interface to the Automatic Message Accounting Recording Center (AMARC) and Engineering and Administrative Data Acquisition System (EADAS) applications. This constrained the freedom to choose interface specifications freely. We wanted flexibility in our design so various combinations of these applications could be configured without trouble or extra cost.

A secondary goal was that as much as possible of the data-link subsystem resulting from our efforts be "portable;" the system should be adaptable to other applications on the 2B, and even other machines, after suitable hardware alteration.

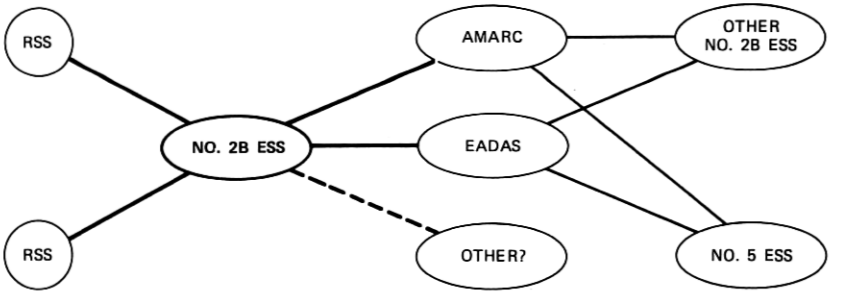
1.2 Boundary conditions

1.2.1 Use of BX.25 protocol

The development of our project proceeded roughly in parallel with the developments of the AMARC and EADAS projects within Bell Laboratories, as well as with the No. 5 ESS project, which was also to interface with AMARC and EADAS. A general consensus was reached to adopt recommendation X.25 of the *Comite Consultatif International Telephonique et Telegraphique (CCITT)*. BX.25, an enhanced subset of X.25, has since become a standard within the Bell System. Its adoption had important consequences in the total design of a data communications system, as will become clear below.

1.2.2 10A Remote Switching System

The 10A Remote Switching System (RSS) is a small space-division system intended for rural communities with fewer than 2000 subscribers. It requires a larger *host* machine to switch interoffice calls. Its controlling program is resident in firmware, and has been developed



AMARC – AUTOMATIC MESSAGE ACCOUNTING RECORDING CENTER
EADAS – ENGINEERING AND ADMINISTRATIVE DATA ACQUISITION SYSTEM
RSS – REMOTE SWITCHING SYSTEM

Fig. 1—Global view of planned No. 2B ESS.

over several years in conjunction with the No. 1/1A ESS, which provided host capability. The 10A RSS feature of the 2BE3 generic program will allow the 2B ESS to be a host as well. Up to 31 RSS machines can be supported by one 2B ESS.

The 10A RSS, having been developed before BX.25, constrained the generality of the data-link system. Nearly 100 units were in service as the 2BE3 generic was being completed. We were required to interface to the RSS with no changes to its firmware.

II. GLOBAL SYSTEM ARCHITECTURE

We address in this section the architectural issues, both hardware and software, involved in the next level of detail from Fig. 1. For this purpose it may be useful for some readers to review briefly the structure of the BX.25 protocol.

2.1 Overview of BX.25

BX.25 is a *layered* protocol consisting of independent layers or *levels* (see Fig. 2). Each level at one side of a data link communicates (in a logical sense) with the corresponding level on the other side. We use three levels, not counting the application program. A call-processing program (say) believes it is conversing with a billing program, for example, at the AMARC. To send a *message* to its counterpart the program simply calls a protocol level 3 interface routine. For its part, the conversation is complete.

Level 3 software will break the message, if necessary, into *packets* of length appropriate to the implementation (depending for practical reasons on the kind of transmission facility, available buffering capacity, etc.). It will then send the packets to the level 3 software at our destination, expecting—as one of its standard functions—a "message-

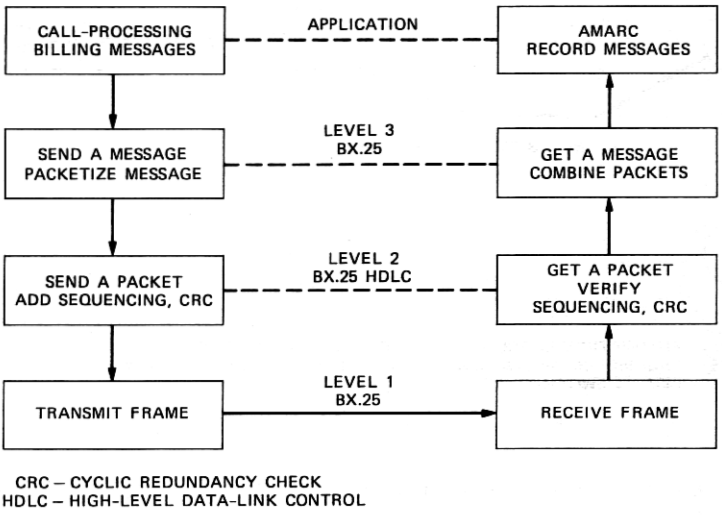


Fig. 2—Application layer and three layers of BX.25.

received" *acknowledgment*. It normally will not rest easy until such acknowledgment is forthcoming within a timed interval; it typically will retransmit the packets until a further interval expires, when it will begin recovery actions. Note, however, that "sending the packets" simply means handing them off to level 2 software.

Level 2 software is very analogous to level 3 in that suitable numbering is attached to the data, which are then delivered as *frames* to level 1.

Level 1 is implemented directly in hardware and is responsible for physically transmitting the data to the remote site. Level 2 frames are specified to contain a Cyclic Redundancy Check (CRC) at their trailing end, though in practice this is computed and inserted by an industry standard hardware chip. Thus there is a slight blurring of the line between levels 2 and 1.

While level 1 is supposed to transmit the data, the primary task of level 2 is to ensure that they arrive safely. Level 2 will retransmit unacknowledged frames, occasionally transmit *receiver-ready* messages on a temporarily unused link ("idle line assurance"), and handle initialization and disconnection functions. The numbering and acknowledgment functions of level 3 are primarily for flow control rather than verification purposes. A primary function of level 3 not yet mentioned is the routing of *logical channels* to their proper physical destinations.

Readers interested in more details about this protocol are referred to Refs. 1 through 3.

2.2 Architectures considered

One possible configuration would be to attach data-transmission hardware directly to the 2B ESS and write level 3 and level 2 software for the ESS. All the detail work of keeping track of unacknowledged frames, and time-consuming interrupts from transmission and reception, would be incurred by the switching machine itself. Since such overhead increases with the number of data links, real-time constraints made this an unrealistic option. In particular, we note that the link traffic increases just at least opportune times—when the machine is busy servicing telephone calls and has the least time available to handle link work. This configuration was never considered seriously. All realistic options use a separate autonomous processor to off-load the ESS.

We will discuss three such options given serious consideration. In the first option, level 3 and level 2 software are wedded together within a small auxiliary processor also containing the data-link hardware. In the second option, level 3 software resides with level 2 software capable of handling multiple data links, many of which are installed on the separate processor. The third option splits levels 3 and 2 into the ESS and attached processor, respectively, with level 2 servicing only one data link. This last option was eventually adopted.

2.2.1 Monogamous levels 3 and 2 together

A configuration with levels 3 and 2 residing in a microprocessor was chosen by the designers of the AMARC side of our data network for their "AMARC Protocol Converter" (APC). This choice has the advantage that level 3 as well as level 2 is off-loaded from the main processor. The disadvantage of concern to us is that when it is necessary to change traffic to the standby link in a multi-link configuration, all level 3 current parameters as well as *all data currently queued at level 3* must be unloaded from one device and transferred to another. At the AMARC side of the link this is not as serious a matter as on the ESS side, where large volumes of data may be generated.

2.2.2 Polygamous level 3 with multiple level 2's

A data link device for up to 16 independent links controlled by common level 3 and level 2 software is currently in use with the Bell System No. 1/1A ESS. This device, named the Peripheral Unit Controller (PUC), was the original model for our own system. Such a configuration has the advantage that links can be changed without intervention of the main processor. However, the PUC turned out not to be suitable for a machine the size of the 2B ESS; with enough power to drive 16 links there is a mismatch in the capacities and costs of the 2B ESS and the data-link controller. This consideration changed the

original direction of our project away from the PUC. Our device was originally to have been a PUC modified from parallel communication with the host to high-speed serial (6.67 Mhz); hence the name Serial Peripheral Unit Controller/Data Link, or SPUC/DL.

2.2.3 Levels 3 and 2 living separately

To answer the previous objection to high overhead cost for the office with few data links, we wished a small and inexpensive basic hardware unit. Vulnerability to failures could then be countered by duplicating units on applications requiring high reliability. An individual link need only be enough more reliable than the communications channel not to add significantly to the failure rate. Furthermore, difficulties in changing links could be reduced by keeping level 3 of the protocol and level 3 buffers within the ESS, which is fully duplicated, including all writable memory. In addition, where a controller implementing Level 3 would almost certainly need to be duplicated and have elaborate checking and matching circuitry to meet reliability requirements, the controller implementing Level 2 can use a simpler reliability check such as parity. Finally, of the three configurations discussed, this last seemed to us to be the most flexible, as well as the most fully portable to other applications and/or hosts. With this scheme substantial new code needed to be written for the 2B ESS—the level 3 protocol, and the maintenance software for the links. Furthermore, complete off-loading of the ESS processor does not occur. Though most of the work and interrupt processing is done in the SPUC/DL, there will be a small traffic-dependent load for managing level 3.

2.3 Configurations by application

As we mentioned previously, each of the three initial applications for the SPUC/DL had its own requirements and the simple hardware could be configured separately for each.

2.3.1 10A RSS

Data links for each RSS are fully duplicated with hard-wired dedicated lines for both links, thus providing a "hot" spare. It is recommended that diverse transmission facilities be used for each data link. These rather strong requirements underscore the high reliability expected of this application. Call processing for lines connected to the 10A RSS is done on the host machine, in our case the 2B ESS. The data links are used to control the progress of calls. Thus, loss of data-link capability means loss of most service for RSS customers. (The RSS can be configured to provide local calls if the host connection becomes lost.) The 10A RSS machine is described at length in Ref. 4.

2.3.2 AMARC

Traffic on the AMARC data links does not carry information that affects customer service, and therefore does not have the intrinsically stringent reliability requirements imposed on RSS links. However, the billing information carried is valuable, and long outages could be costly. Short outages—some tens of seconds—would be tolerable if suitable buffering capacity were provided to avoid losing billing data. AMARC links were therefore configured with one data link on a dedicated hard-wired line, supplemented with a "cold" spare connected to the switched network. Changing to the cold spare after a dedicated primary link outage exceeding some minimum duration involves dialed telephone calls between the ESS and AMARC machines. Such a scheme saves the considerable cost of a second dedicated connection.

2.3.3 EADAS

Though data gathered on EADAS links are useful for planning telephone network growth, their temporary loss does not compromise either customer service or revenue. EADAS applications can therefore be configured with a single data link, equipped with a dedicated line.

2.4 Architectural summary

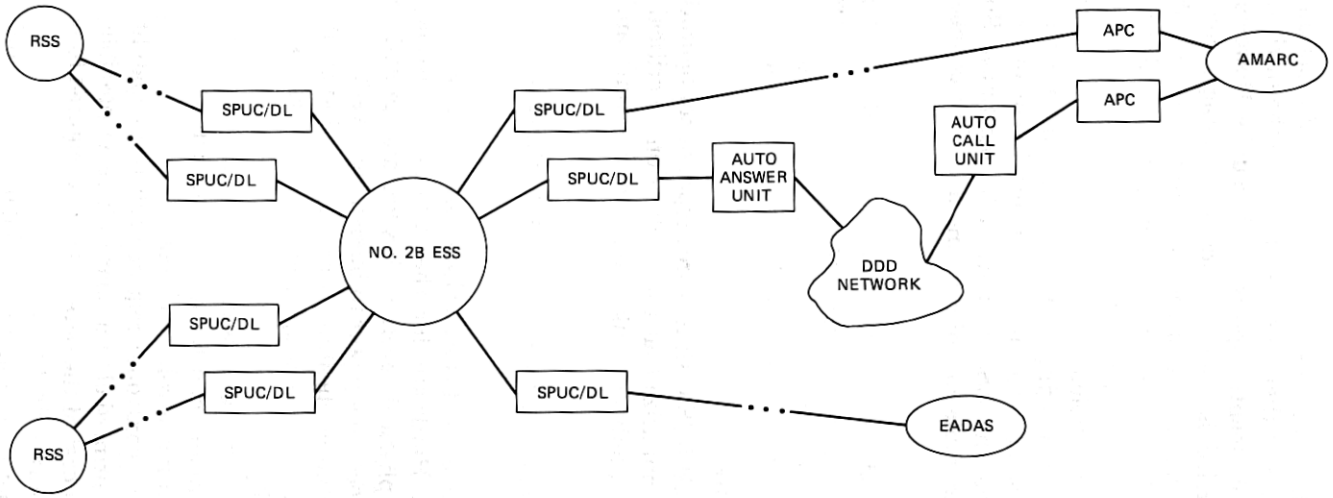
A diagram of a No. 2B ESS with its three types of configurations is shown in Fig. 3. We emphasize that the duplication on critical applications is for reliability; only one link is active at any time, and there is no load sharing.

III. FUNCTIONAL REQUIREMENTS OF THE SYSTEM COMPONENTS

3.1 Hardware

The general requirements of the hardware are as follows:

1. Basic hardware. Hardware for the data-link controller must execute level 1 (the physical level) of BX.25, and provide enough memory space and processing power to implement level 2.
2. Link speeds. Initial applications were to use 2400 or 4800 bits/second (bps), but for possible new applications a goal was set for 56K-bps hardware capability.
3. Link Reliability. Our goal was to have data communication failures be caused predominantly by transmission facility failures rather than by data-link controller failures.
4. Repairability. The controller required self-test capability to isolate faults to a single-printed circuit board wherever possible.
5. Cost. The initial cost of the data links was required to be low for smaller offices with only a few links.
6. Expansion. Expansion should be modular, smooth, and trouble-free.



AMARC – AUTOMATIC MESSAGE ACCOUNTING RECORDING CENTER
APC – AMARC PROTOCOL CONVERTER
DDD – DIRECT DISTANCE DIALING
EADAS – ENGINEERING AND ADMINISTRATIVE DATA ACQUISITION SYSTEM
RSS – REMOTE SWITCHING SYSTEM
SPUC/DL – SERIAL PERIPHERAL UNIT CONTROLLER/DATA LINK

Fig. 3—The three data-link applications of the 2B ESS.

3.2 Software for the SPUC/DL

Stated simply, the SPUC/DL software requirement was to implement level 2 of the data-link protocol. Unfortunately, the three data-link applications for 2BE3 required two sets of protocol rules for level 2. The AMARC and EADAS applications use the procedures defined by the BX.25 specifications,[3] which were derived from the CCITT X.25 "Link Access Procedure B" or LAPB. The third application, RSS, uses the older version of the X.25 level 2 procedure "LAP." LAP and LAPB, though similar, are not compatible (LAP cannot communicate with LAPB). However, except for the initial handshake, some error recovery procedures, and a few differences in command formats, the core of information processing procedures is the same.

Initial SPUC/DL applications were to use link speeds of 2400 bps, although future applications of AMARC requiring 4800 bps were foreseen. These requirements can be easily met by a microprocessor device.

Data traveling over common carrier facilities are inherently susceptible to any noise disturbances afflicting such facilities. It is essential that data altered during transmission be detected and the correct data retransmitted. This is the *raison d'etre* for level 2 of X.25. To this end, X.25 employs a 16-bit Cyclic Redundancy Check (CRC) to detect frames altered by noise, and a sequence numbering scheme to facilitate retransmission of frames discarded because of bad CRC. Flow control procedures at level 2 also prevent data loss from buffer overflow.

An additional requirement for maintenance purposes was for some diagnostic capability "on board" the SPUC/DL. The on-board (resides in SPUC/DL memory) diagnostic was to be used by the ESS diagnostic to test hardware inaccessible to the ESS or requiring excessive ESS resources to test. When the ESS wished the SPUC/DL to begin its diagnostic routines, it was to instruct the SPUC/DL operating system to relinquish control to the on-board diagnostic programs. To go back to normal the ESS must issue a master reset of the SPUC/DL, since some diagnostic procedures would destroy the normal operating state.

Diagnostic requirements include a "loop around" test in which data packets would be sent out on the link, directly echoed back by the remote end, and received by the SPUC/DL and 2B ESS. This would normally be a protocol violation in the use of an *address* field at the frame level. We have enhanced the implementation to suspend normal protocol address checking when the diagnostic is running.

3.3 Software within the 2B ESS

We discuss here our requirements for both the protocol software and the maintenance, or fault-tolerant, software. Most of the protocol

requirements are general and serve to elaborate on our previous discussion of BX.25.

3.3.1 Protocol-implementing software

The 2BE3 generic program requires dedicated point-to-point (permanent virtual circuits), full-duplex, 2400 (or 4800) bits per second communication facilities. The No. 2B ESS software implements only those portions of the BX.25 specification needed to meet the 2BE3 generic requirements.

Level 3 of the BX.25 protocol controls the transfer of data between level 2 and the next higher level (the application layer in No. 2B ESS). Transferring includes several processes, namely: packetization, multiplexing, sequencing, and flow control.

3.3.1.1 Packetization. Packetization is the process of a transmitter's dividing large messages into practical-sized packets, transmitting the packets, and having the receiver concatenate the packets to form the original messages. BX.25 does this by setting a "more-data" (M) flag bit in the level 3 header in all but the last packet of a message. The AMARC billing data messages have a maximum of 512 bytes; EADAS thirty-minute traffic data messages vary, with an average of 4,000 bytes depending on the central office. The packet size of 256 bytes was determined appropriate for the throughput desired and the expected error rate of the data transmission facility. If the packet size is large and the transmission facility error rate is high, many packets may receive errors and need to be retransmitted, which decreases the throughput.

3.3.1.2 Multiplexing. Multiplexing is the process whereby a transmitter combines data packets with identical destinations but associated with various application functions (logical channels). The receiver separates (demultiplexes) the packets via the logical channel numbers and distributes them to the appropriate application function. AMARC has a billing-data logical channel, EADAS a traffic-data logical channel, and both applications have a time-of-day logical channel; thus, multiplexing is required with both the AMARC and EADAS applications.

3.3.1.3 Sequencing. Sequencing of BX.25 level 3 data packets is required to assure packets are received in the identical sequential order of transmission. Each level 3 data packet header contains two modulo eight sequence numbers: a send sequence number P(S), which identifies the data packet itself, and a receive sequence number P(R), the expected number of the next data packet received.

BX.25 permits a level 3 transmitter to send a prescribed number of packets to level 2 without obtaining acknowledgment that earlier

packets were received correctly. These unacknowledged level 3 packets sent to level 2 are in the *window* until their reception has been acknowledged. The *window size* of each logical channel is the number of packets that can be sent to level 2 without being acknowledged; the AMARC window sizes are both five while EADAS windows are both two.

3.3.1.4 Flow control. Flow control is provided by BX.25 to permit receivers a method to regulate the rate at which data packets are sent to it. A packet received with send sequence number n is acknowledged with the next packet sent by setting the receive sequence number to $n + 1$. Any receiver having a data-capacity problem may delay the acknowledgment of received packets and avoid data loss.

3.3.1.5 Requirements for the 10A RSS. RSS level 3 was originated with No. 1 ESS before the acceptance of X.25 and is not compatible with BX.25. RSS level 3 messages contain a unique "SYNC word" in the first two bytes. Received messages are discarded if they do not begin with a SYNC word. The SYNC word is manipulated entirely by the RSS level 3 program and is transparent to the No. 2B ESS application call-processing software.

A two-byte remote terminal header following each SYNC word contains the RSS message type in a five-bit "client identity" field. The level 3 software distributes (demultiplexes) received messages to the call-processing application programs by interpreting the client identity. An RSS level 3 header also contains an eight-bit word count field specifying the number of data words in the message.

Packetization of messages is also used in RSS level 3 communication. The maximum RSS packet size is sixteen bytes and the packet boundary is totally asynchronous with messages from an RSS. RSS-bound messages transmitted from the No. 2B ESS always begin a new packet but may require several packets, depending on their length.

Communication concerning only the No. 2B ESS and SPUC/DLs is passed in *control packets* with a unique packet descriptor. A *packet descriptor* is the first two bytes of *all* packets passed between the 2B ESS and a SPUC/DL, identifying the packet type (data or control) and specifying the packet length in bytes. Packet descriptors are not transmitted over the data links. Control packets are external to level 3 and pass directly between the SPUC/DL and its maintenance configuration and diagnostic programs.

3.3.1.6 Timing requirements. We have mentioned previously that the decision to put level 3 in the 2B ESS imposes a penalty in traffic-dependent processing power. In requirements terms, level 3 had to be fast.

3.3.1.7 Other requirements. It was foreseen that part of the protocol-handling software package would include low-level routines for physical I/O to each SPUC/DL. Two requirements were imposed on these I/O routines: error routing and monitoring. Packets whose reception was not completely normal were to be routed to the data-link maintenance software. This includes nondata (control) packets, I/O errors, and packets from a nonactive SPUC/DL. A manually invoked monitoring utility was to report part or all the data sent to and read from an application. The monitor proved to be of value during system debugging and integration.

3.3.2 Maintenance (fault-tolerant) software

There were several general requirements assigned to the data-link maintenance function.

1. Initialization and Loading. On certain rare events there may be occasion to reinitialize any or all links in the system. This may require a down-load of the SPUC/DL program from its magnetic-tape residence at the ESS. This would also be done when field-updating SPUC/DL software.
2. Handling Error Reports. Maintenance software is required to maintain records of such reports, which may include frequent loss of facility carrier, low efficiency of transmission, etc.
3. Error Recovery/Link Reconfiguration. Serious faults (e.g., loss of communication with a SPUC/DL) or slow degeneration of a link may require removal of the affected link from service and changing of traffic to the standby link, if available. Thus, data-link maintenance is charged with trying to reconfigure a failing system to maintain traffic.
4. Regulation of the Diagnostic Program. The diagnostic, controlled by the maintenance software, is to be executed automatically when a failure occurs, and routinely on a nightly basis to detect faults on standby links before they are needed in service.
5. Audits. Maintenance "audit" programs were required to check periodically the legitimacy of the control-block database of data-link maintenance, to correct and report single errors, and to reinitialize if multiple errors were found.
6. Console Operator Interface. The final requirement of the data-link maintenance software was to interface with the human operator of the system, providing keyboard input service and output of various responses, as well as spontaneous reports of normal and abnormal events.

3.3.3 The diagnostic program

The diagnostic program had to detect and locate faults in the 2B

processor I/O interface, the SPUC/DL, and the component parts of the transmission facility. In the RSS application, it must also locate faults in the remote data-link control hardware. The diagnostic control structure was required to be sufficiently flexible to permit systematic testing during the installation of the SPUC/DL, the near- and far-end data facilities, and (for RSS) the remote data-link equipment. It had to be able to stress the overall data link enough to recreate faults observable only during heavy traffic.

IV. DESIGN AND IMPLEMENTATION—SPUC/DL HARDWARE AND SOFTWARE

4.1 SPUC/DL hardware

We can identify three key decisions in the evolution of our design. All three involved hardware/software trade-offs, the software being in the 2B ESS. The first decision involved the scope of the SPUC/DL, as we have discussed in Section II. The second decision was the use of periodic polling of the SPUC/DL rather than program-interrupt hardware. We will elaborate on this in Section V. The third decision was to use writable Random Access Memory (RAM) instead of Read-Only Memory (ROM).

4.1.1 Use of writable memory

The advantage of RAM is that updates of SPUC/DL software for circuits in the field are much less costly, requiring only software procedures rather than replacement of physical chips. The main disadvantage is that the memory is volatile; the program is destroyed (and must be reloaded by the 2B ESS) if power to the SPUC/DL fails; therefore, software within the 2B ESS had to be developed for downloading. In addition, the SPUC/DL software must have a nonvolatile residence from which it can be loaded. In our case this was to be the magnetic tape cartridge of the 2B ESS. The need for loading would cause concern only if long delays were incurred by the loading process. However, since 2B ESS-SPUC/DL communication proceeds nearly at memory speeds, the loading time mostly comprises the time it takes to retrieve the program from tape.

4.1.2 Logical design

The data-link controller itself consists of a single 16-bit microprocessor, its associated RAM, and two Direct-Memory Access (DMA) controllers, DMA1 and DMA2. Duplication within a single SPUC/DL is not necessary because of the relaxed reliability the distributed architecture allows. DMA controller 1 is a multiple-channel device allowing data transfers to take place between the 2B ESS and the

SPUC/DL independently of the microprocessor. Similarly, DMA controller 2 provides independent transfers between the SPUC/DL and the transmission facility (see Fig. 4). This architecture minimizes demands on the microprocessor while allowing high throughput. Level 1 protocol is incorporated in a Universal Synchronous-Asynchronous Receiver-Transmitter (USART) and standard interface chips.

4.1.3 Physical design

To meet the requirements of modular growth and high multiple-

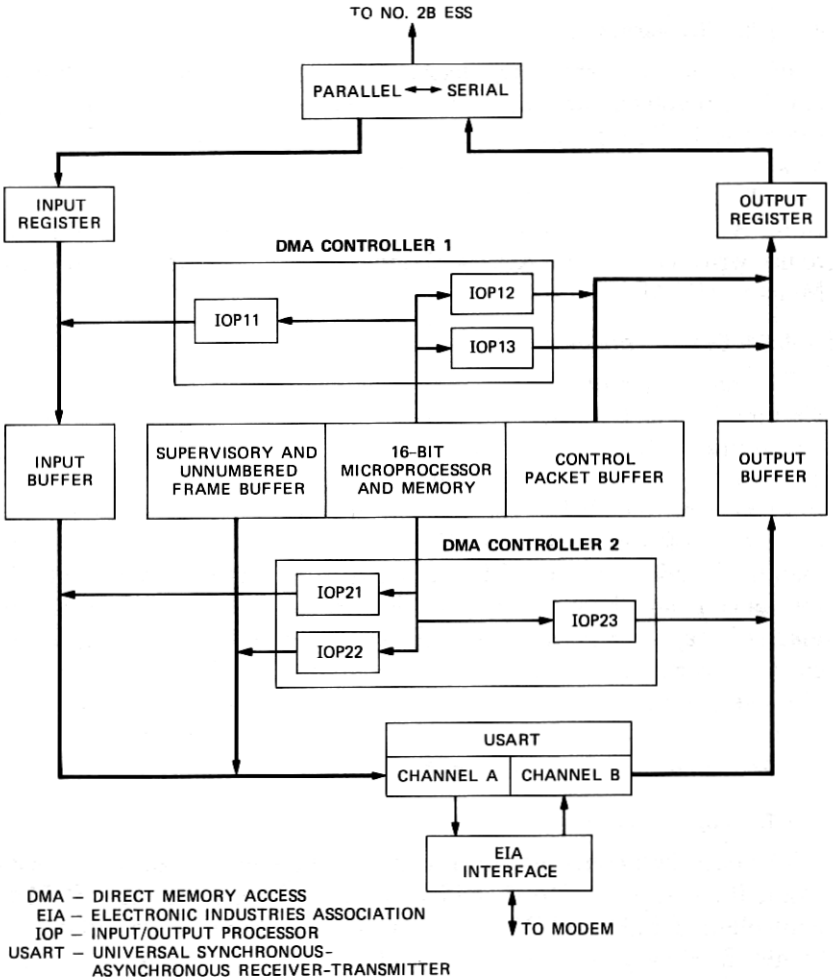


Fig. 4—SPUC/DL hardware components and architecture.

link reliability, each data-link controller was made a separate circuit. Pairs of controllers share a −48 volt power lead. Each SPUC/DL has its own power supply, transmission facility interface, microprocessor, memory, and communications channel to the 2B ESS. There are four printed circuit boards per SPUC/DL, containing the microprocessor and associated logic, memory, the data communications hardware, and the logic for I/O to the 2B ESS. Four complete SPUC/DL circuits, their associated power supplies, fuses, and connectors are housed in a single chassis.

A concentrated effort was made to provide a straightforward maintenance procedure. Each circuit has its own power fuse and modem cable. Each circuit is numbered, and this numbering is used on the circuit board enclosure, power supply housing, power fuse, and modem connector. In addition, each circuit is color coded, and the identifying color appears everywhere the circuit number is used. The 2B ESS SPUC/DL diagnostic program prints both the circuit name and color when it refers to a particular SPUC/DL.

4.1.4 Reliability issues

To provide sufficient reliability, the data-link controller's memory (program and data) is stored with parity and the parity is checked whenever data are read out or written. In addition, write-protection hardware is provided for the program memory. Any attempt to write into program memory will be prevented, and the 2B ESS will be informed. The status of the SPUC/DL hardware is summarized in a hardware status register that the 2B ESS can read.

The SPUC/DL on-board diagnostic program first computes a check sum over the program memory, tests the data memory, and then confirms proper operation of the microprocessor itself. Once the basic sanity of the microprocessor and correctness of its program is established, the diagnostic program goes on to test both the DMA controller that oversees data-link communication and the USART. The USART is tested by actually transmitting on the link. Additional loop-around circuits at the EIA interface, the associated modem, and the far-end modem allow faults to be isolated and identified easily. This philosophy of testing sections of the controller nearest to the central processor and then relying on those sections to test sections further removed allows economical and rapid fault isolation. Fault isolation is enhanced by partitioning the SPUC/DL into its four circuit boards. All interface circuitry to the 2B ESS is on one board. The second board contains DMA controller 1 and address decoding circuitry. The third board contains all data and program memory. The fourth board has the microprocessor itself, DMA controller 2, and the USART and modem

interface. Since the proper functioning of each board relies on the previous board but not on subsequent boards, most faults can be isolated to a single board.

4.2 Software within the SPUC/DL

Because of the commonality between LAP and LAPB, it was decided to implement both of them in one software load; the ESS would specify one or the other with a parameter sent to the SPUC/DL on initialization. Though this decision increased the space used for code in the SPUC/DL, it eliminated keeping track of which code to download into which SPUC/DL and the need for extra space on the cartridge tape. Separate codes would also have caused complications with the diagnostic. Perhaps the largest factor in the single-program decision was the expense of having to design, code, test, and administer two separate programs.

First we look at the SPUC/DL software structure from a functional data flow viewpoint, and then at the overall structure of the operating system that implements these functions.

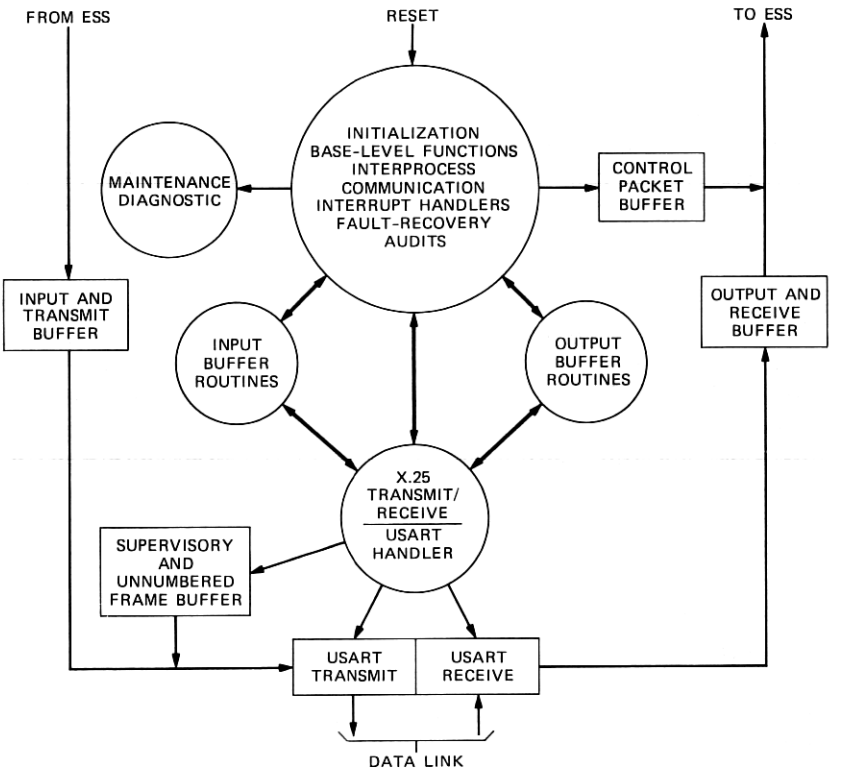
4.2.1 Data flow within the SPUC/DL

The hardware architecture of the SPUC/DL shown in Fig. 4 closely parallels the flow of data through the device. Internal to the SPUC/DL the send and receive processes run simultaneously. Data packets from the ESS enter the input buffer under control of DMA1 and are cataloged by the input routines, with *control* packets (messages for the SPUC/DL itself rather than for transmission) being processed immediately. Once packets have been cataloged, they are processed by the X.25 routines, which assign sequence numbers and set up the DMA2 transfer to the USART for output over the link as X.25 frames. Simultaneously with transmission, frames are coming in from the receive channel of the USART, are processed by the X.25 routines, and pointers to information frames are passed to the output routines. The output routines add packet descriptors to the information frames, and then set up the DMA transfer to the ESS.

This is a high-level view of the SPUC/DL and there are many other software processes involved in its operation, including maintenance routines, which play a large role in support of data handling.

4.2.2 SPUC/DL Operating System

The SPUC/DL Operating System may be visualized as a hierarchical multiprocessor operating system. In this system the 16-bit main processor acts as a master while the six individually programmable DMA channels are slave processors. Once a DMA channel has been



ESS – ELECTRONIC SWITCHING SYSTEM
USART – UNIVERSAL SYNCHRONOUS-ASYNCHRONOUS RECEIVER-TRANSMITTER

Fig. 5—SPUC/DL software processes; interprocess communication has been emphasized.

given a process, it will run that process to completion without further attention from the master. We denote these single channels as I/O processors, or IOPs. Most interprocessor communication is one way between the master and the IOP. Interprocess communication is handled using producer/consumer semaphores.

Figure 5 shows how the SPUC/DL software processes use the hardware to carry out the data communications function. Each I/O processor handles a specific I/O function. Communication between the SPUC/DL and ESS is handled by three channels of DMA1. In Fig. 4 they are labeled IOP11, IOP12, and IOP13. IOP11 runs the process of moving data from the input register to the input buffer. Once the SPUC/DL has been initialized by a master reset, the master processor starts this process running on IOP11. This process runs continuously and becomes inactive only when input buffer space is

exhausted or the process may be preempted by the microprocessor for status updates. This status information consists of "where will the next word be placed in the input buffer," or "deallocate *n* input buffer words." Unlike IOP11, IOP12 and IOP13 run only when there are processes scheduled for them (packets to be sent to ESS). IOP12 transfers control packets from scratch memory to the output register and IOP13 transfers data packets from the output buffer to the output register.

The DMA2 processors have almost mirror-image functions. IOP21 moves packets from the input buffer to the USART where they are serialized and sent out over the link. Packets are scheduled for IOP21 on a first-in/first-out (FIFO) basis with priority for packets being retransmitted. IOP22 moves supervisory and unnumbered frames from scratch memory to the USART. IOP23, which moves data in from the link, does not run continuously because of the intervention of the master processor to handle alignment of the start of incoming data. All other processes are run by the master processor, including the considerable primary task of processing X.25 frames.

4.2.3 SPUC/DL software reliability

In developing SPUC/DL programs considerable attention was given to reliable software design through modularity. Both the SPUC/DL Operating System and the level 2 procedures were coded in the C language and a top-down design structure was used wherever practical. Assembler-level code was used only where necessary for speed or to handle hardware-dependent functions. The high-level language had advantages both from an implementation and a testing standpoint. However, because both the hardware and software were new to us, during debugging it was occasionally necessary to look at the assembly code or binary output of the C compiler to determine the source of some problems.

The hardware of the SPUC/DL provided three features that also enhanced software reliability. First, the use of DMA hardware controllers for I/O eliminated the need for complex scheduling algorithms to move blocks of data. This is especially important in an I/O-intensive application such as a data-link controller. Second, write-protection hardware was provided. Since the program store of the SPUC/DL is in writable memory, without the write-protection feature of the hardware, the program would be vulnerable to bugs that alter the program itself. If a write-protect violation occurs, the SPUC/DL software is not altered but an interrupt routine attempts to save as much pertinent information as possible and then proceeds normally. The ESS is notified by a message from the SPUC/DL containing the hardware status register. The third hardware feature is a "sanity" timer, which

expires unless periodically serviced by the software. The timer guarantees that the microprocessor will not get stuck in any one place for an inordinate amount of time. If the sanity timer should expire, an interrupt is generated and is handled similarly to the write-protect error.

In addition to the hardware mechanisms mentioned, reliability is increased by periodic software audits. The audits check major global data structures, looking for inconsistent or out-of-range values. No attempt is made to correct audit-detected errors. The event is reported to the ESS, which prints a message.

4.3 System performance

The SPUC/DL software has an average one-way delay of about 4.5 ms. It has been tested at 5 to 10 times the speed expected in initial field use.

V. 2B ESS HOST SOFTWARE

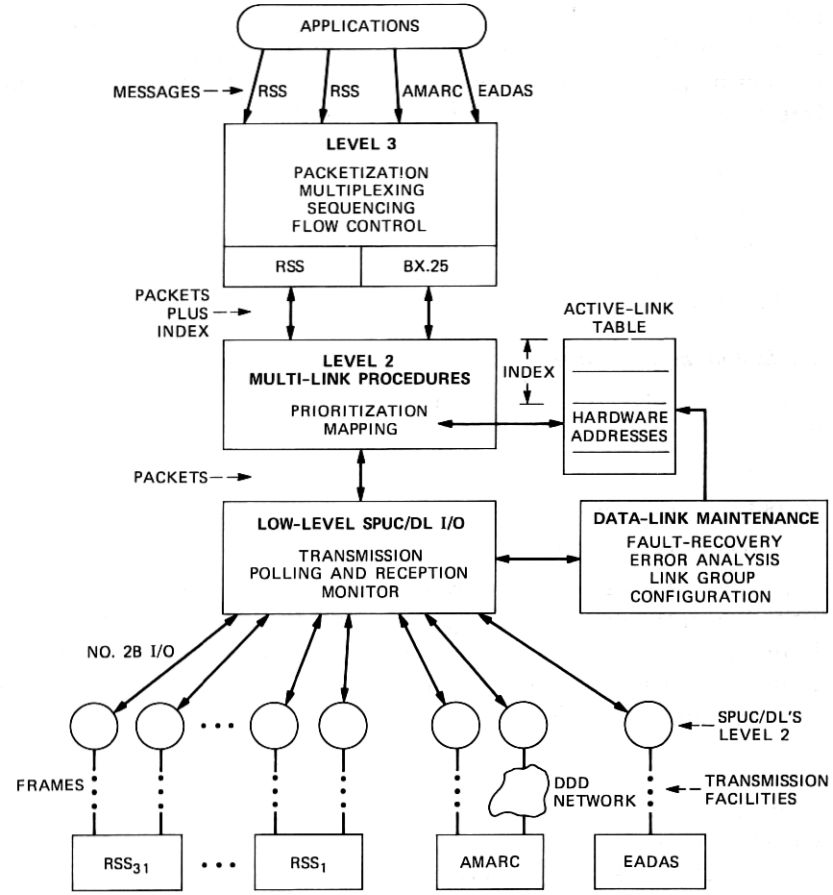
5.1 Architectural overview

Figure 6 is a high-level diagram of the data-link software in the 2B ESS. When they have a message to transmit, call-processing routines, billing routines, and data-collection routines enter appropriate points in level 3. Level 3 converts the messages into packets, breaking large messages into multiple pieces if necessary. Multiple logical channels (logical destinations) are sorted for their physical destination (multiplexing). Appropriate sequence numbers are attached. The packets are transmitted immediately, if possible, or put on a queue if flow is currently under control.

When the packets are ready to be transmitted, the software passes them to the level 2 Multi-Link-Procedures. These routines sort the packets by priority, and then do a *mapping* to a definite hardware address (i.e., physical SPUC/DL). The mapping is done through a table managed by the data-link maintenance software. Thus, the routines doing transmission need have no knowledge of which link they are using.

The Multi-Link-Procedures pass the packets to the low-level transmission software, which physically outputs them to the SPUC/DL whose hardware address was passed as an argument. The link-traffic-monitoring utility is implemented at this level. Level 2 in the SPUC/DL ensures error-free transmission to the destination.

The inverse process, packet reception and assembly into messages, is similar. Note that on reception, not all packets get routed to the Multi-Link-Procedures. Control packets originating within the SPUC/DL, and information packets from links not currently active are given to the data-link maintenance software. The maintenance



AMARC – AUTOMATIC MESSAGE ACCOUNTING RECORDING CENTER
DDD – DIRECT DISTANCE DIALING
EADAS – ENGINEERING AND ADMINISTRATIVE DATA ACQUISITION SYSTEM
RSS – REMOTE SWITCHING SYSTEM
SPUC/DL – SERIAL PERIPHERAL UNIT CONTROLLER/DATA LINK

Fig. 6—Data-link software within the No. 2B ESS.

software also uses the low-level common I/O routines to transact its business with each SPUC/DL. In this way the monitoring utility will report *all* communication to any link or link group.

5.2 Device polling and timing considerations

To have the SPUC/DL request attention from the 2B ESS, a polling method was adopted owing to complications in the 2B ESS demand-interrupt hardware. When it requires attention from the 2B ESS, the SPUC/DL sets a bit in its status register. The status register's bit controls a *ferrod*, a ferrite device whose magnetic state may be sensed

under program control by a *scanner*. These particular ferrods are arranged in rows of 16, so the status of 16 SPUC/DL "packet-present" ferrods may be read simultaneously. Once every base-level loop (100 milliseconds) the 2B ESS will poll all scanner rows containing SPUC/DL packet-present ferrods.

Clearly, such polling is inefficient; for polls on which no SPUC/DL has data, the polling time is wasted. The total time available for processing calls is reduced, and so therefore is the performance of the 2B ESS. The actual time penalty depends on how fast the software is. In our case the loss is around 0.6 percent with idle data links.

Of greater concern are the effects of measured-service billing (billing every telephone call) to the AMARC, and of RSS data-link traffic (several messages are required per call). In the AMARC case, the data links shorten the time spent on billing by about 30 percent relative to the previous use of magnetic tape. In the RSS case, only around 7 percent of the real-time of a typical call is due to the presence of the SPUC/DL. We can conclude that level 3 processing and programmed I/O to each SPUC/DL are not serious drains on the time available to the ESS, relative to its other tasks.

5.3 Maintenance software

The link(s) from the 2B ESS to any single remote site make up a *link group* of one or two links. It is the task of data-link maintenance to ensure that a link is always active in every link group.

5.3.1 Architecture

The highest-level component of data-link maintenance is called *link-group control*, and has responsibility for keeping a link always active in every group. Because of the substantial differences in the three main applications, link group control was divided into three independent sections, one handling the EADAS link group, one handling the AMARC link group, and one handling all RSS link groups.

Link-group control depends on a collection of utility routines called *link-state control*, which performs tasks on single links, independent of the type of link group of which the link is a member. Conceptually, a given link may be thought of as controlled by, or belonging to, either link-group control or link-state control (but not both) at any one moment. Link-group control manages links that are "in service," that is, actively carrying data ("active") or available for such use if needed ("standby"). Link-state control operates on links that are "out of service" by virtue of having been "removed"—either automatically after detection of a fault, or manually through issuance of a console keyboard command by the operator of the system.

The link-state-control routines depend in turn on a *loader* for down-

loading the SPUC/DL software from its residence on the system magnetic tape cartridge, and on the diagnostic program, which is described in more detail below.

The software of the 2B ESS continuously executes a fundamental "base-level loop" of some 100 milliseconds. Within the loop various functions are given control with the understanding that no function will exceed its own allotment of time. That is, instead of being driven by a clock interrupt, context switches (task changes) are the responsibility of the base-level routines themselves. They must save and restore their own contexts if completion of their functions requires more than one allotment of loop time. For this purpose and for keeping longer-term records, data-link maintenance makes use of control blocks.

5.3.2 Data structures

Two main types of control areas are maintained by the software. The link-group control keeps a link-group control block for each link group. In addition to current-function context information for link-group control, records are maintained of the number of changeovers between links in the group. This can prevent thrashing back and forth during periods of trouble, such as a lightning storm. In general, this block is intended for those records associated with the entire group, rather than with one specific link.

The second main type of control block is maintained by the link-state control routines, and is called a link-state control block. One such block is reserved for every link defined in the 2B ESS database. Here is stored information on the current context of link-state control for each link. In addition, a collection of records is maintained about the recent history of each link. For example, it is remembered at what time the diagnostic program was last run (to prevent too frequent use of the diagnostic), as well as its result. Over 30 fields are reserved for counting each of the types of errors reported by the SPUC/DL or recognized by the 2B ESS. Cumulative counts are also kept for each of several classes of errors, a class being defined by its relative seriousness.

The other major data structures used by data-link maintenance are the permanent (read-only) office records defining the data links present and their attributes (modem type, hardware address, etc.) These structures are accessed through special database routines.

5.3.3 Recovery strategies

The primary strategy for recovering from data-link failures is to use the redundancy engineered into the system—that is, to move link traffic to the other link in a two-link group.

The 2B ESS can detect failures of various natures and degrees of severity. The most obvious and straightforward is an inability to communicate with the SPUC/DL because of failures of hardware input or output orders. Here it is first attempted to change, if possible, to the standby 2B ESS processor to see if the orders can then be executed. If not, the fault is classified as *hard.* This is the fault detected when the power supply of a SPUC/DL fails (or is turned off).

A wider range of problems can be discovered by the checksum mechanism. During initialization sequences and when returning an out-of-service link to service, the ESS resets the SPUC/DL by setting a bit in the status register. This action forces the SPUC/DL program counter to an absolute address where its initialization code begins. As part of this code, a checksum is computed on the entire program memory of the SPUC/DL. The computed checksum is returned to ESS along with a prestored value it is supposed to match. The return of a matching checksum shows that the ESS can access the SPUC/DL, that the processor within the SPUC/DL is functioning, that the program is loaded and is probably correct, and that the device is in a known state. A bad checksum probably means a memory fault. In an emergency, a link can be "forced" into service even with a bad checksum comparison, so long as some checksum is returned.

Some symptoms are generally associated with the carrier facility rather than the data link. As mentioned previously, when troubles such as carrier loss are encountered, they are recorded. Noise on the facility will cause low link *efficiency*, defined as the ratio of bytes acknowledged properly to the total number transmitted. These troubles and a variety of others are individually counted within the data-link maintenance software. When the cumulative counts of such errors exceed some threshold value, recovery action begins. The counts of each type of event are weighted by their relative seriousness or impact on the system. The recovery strategy for these (individually) "nonfatal" errors is to compare the recent history of such errors for both links in an RSS link group, changing links if the standby SPUC/DL seems better. In an AMARC link group, since there is no guarantee that a standby link is available at the AMARC site, the 2B ESS must depend on the AMARC to detect facility problems and to change links. Only a severe problem will cause the 2B ESS to remove an AMARC link.

5.3.4 Changing links

The operation of changing traffic from one link to another in a link group is fundamental in the data-link recovery strategy, and required considerable development effort. The two applications for which the

operation is required, AMARC and RSS, are quite different. We will give a brief overview of each.

5.3.4.1 Changing AMARC links. The change operation on an AMARC link group attempts to follow the BX.25 protocol in most respects. Carrier is detected on the standby link after automatic dialing and answering hardware completes the connection via the DDD network. The ESS sees that the AMARC machine issued a "disconnect" on the primary link within its protocol level 2. Next, level 2 of the protocol becomes active on the standby link. Suitable security precautions are now taken to ensure the connection is genuine. Finally, level 3 of the protocol begins, and the operation is complete.

In the other direction the procedure is simpler. After the protocol has initialized at level 2, receipt of any information frame on the primary link will cause the 2B ESS simply to disconnect (hang up) the standby link.

5.3.4.2 Changing RSS links. For historical reasons, changing traffic between RSS links is complicated. The procedure was originally developed for the 1/A ESS. As we described in Section II, in the 1/1A environment both RSS links are connected to a single processor, the PUC/DL. Since there is no level 3 protocol for the RSS, resetting protocol level 2 (such as might be expected when changing links) has serious repercussions. Level 2 resets cause loss of data because they clear frame buffers. In the PUC/DL a changeover procedure was designed so no level 2 reset would occur and no data would be lost. The PUC/DL transfers the outstanding frames and current values of the protocol parameters (sequence numbering) to the new hardware by changing a pointer in its database. Thus, after changing links the old level 2 is running on the new link. Similar actions occur in the RSS; no reset of level 2 takes place, and no data are lost.

Since it was part of our design goal to leave RSS code unaltered, it was necessary for the 2B ESS and SPUC/DL to emulate the actions of the PUC/DL during the changing of the links. This was done by the following sequence: (1) protocol level 2 is brought up on the standby link, though transmission of data is disabled; (2) transmission stops on the active link while it is emptied of received data; (3) untransmitted and transmitted but unacknowledged data are read out of the active link and written into the standby link; (4) the protocol state of the active link is transferred to the standby to preserve the sequence numbering; (5) a "change links" message is transmitted to the RSS on the standby link, which turns the standby into the active link, and causes the RSS to change also.

5.4 Diagnostic software for the SPUC/DL

Fault location for the SPUC/DL data links can be best characterized

by its diversity. The diagnostic must contend with three different link configurations—AMARC, RSS, and EADAS—each having its own testing requirements. There are three execution environments: the 2B ESS, the SPUC/DL microprocessor, and for RSS, the remote terminal microprocessor. In addition, the diagnostic faces unusual conditions within the 2B ESS. It must be executable in the off-line processor, and must accommodate several instances of itself executing in parallel.

We have added two features to the diagnostic that are new to the 2B ESS. First, the SPUC/DL diagnostic specifies explicit replaceable units in the TTY output messages, rather than coded trouble location information. Second, to simplify installation and growth of a SPUC/DL configuration, we have implemented *partial* diagnostic capabilities—the ability, for example, to run only the first few tests so uninstalled hardware will not cause error messages.

To achieve flexibility in dealing with different link configurations, the diagnostic was organized for table-driven execution of independent tests, the table being specified either by the application type, as defined in the 2B ESS database, or by an operator-generated input message. This permitted maximum use of common diagnostic code, yet flexibility in tailoring specific tests for specific applications. The table also provided a simple means of providing special test functions not normally run—for example, an exhaustive SPUC/DL program memory test rather than the checksum test normally used by the routine diagnostic.

The diagnostic proceeds with the 2B ESS executing directly initial interface tests to the SPUC/DL, then requesting the SPUC/DL to run tests of its internal hardware. It then uses the SPUC/DL to do analog loop-back testing to the local modem, followed by digital loop-back to the remote modem, and finally (for RSS) requesting remote terminal hardware tests to be run. For those tests not executed by the 2B ESS, the diagnostic acts as a control program and analyzes data to convert the cryptic test failure information into human-readable messages specifying the replaceable unit.

While most of the diagnostic tests for the SPUC/DL data links use conventional diagnostic testing techniques, the digital (remote) loop attempts to find faults normally discovered under heavy-traffic conditions. This test sends 1000 sixteen-byte packets from the 2B ESS through the SPUC/DL, over the link, through the remote loop-back, and back to the ESS through the SPUC/DL. The SPUC/DL uses its operational link efficiency algorithms to quantify the quality of the link. A link is reported as a good link, a failing link, or a "degraded" link—usable but performing below what is considered optimum.

After a fault has been repaired, it is necessary to test the 2B-SPUC/DL interface from both the on-line and off-line central processors.

This ensures not only that faults in the off-line interface are located but that repair of an interface board has not introduced a fault in the standby processor to SPUC/DL interconnection.

The SPUC/DL diagnostic is the first program to take full advantage of a newly enhanced 2B ESS software subsystem for maintenance, which provides a degree of multitasking convenience. Use of the "Multi (base-level) Scan Function" (MSF) program allows for multiple simultaneous executions of the diagnostic on different data links with a single copy of the diagnostic and several instances of its data memory, one for each execution. The only significant complication is message routing from each SPUC/DL to the appropriate data memory, which is done through interfaces with the I/O and the data-link maintenance programs.

VI. PROJECT POSTMORTEM

6.1 Problem areas during development

Though the difficult problems of the project integration phase are freshest in our minds as this paper is written, there were problems of various kinds throughout the project. The most difficult *technical* problems in the data-link area were met in the integration and final debugging phase of the project. The problems were usually lost words of data, or protocol failures that stopped communication. They occurred randomly in time, usually when data-link traffic was high, and were not easy to reproduce. At least three factors interacted to increase the difficulty in solving the problems.

First, the SPUC/DL test facility did not have enough capacity to stress the links. Only the 2B ESS laboratories could drive the links at the acceptance-test levels, and time in these laboratories was in great demand by people working on all aspects of the 2BE3 generic program. The SPUC/DL test facility was improved, but only after it was clear that serious but infrequent problems were present in the SPUC/DL. The test-facility shortcomings meant that the SPUC/DL had more problems than it should have during the integration phase. The service-affecting problems had been mostly resolved by the time the test facility became really adequate.

The second factor lies in the distributed nature of our systems: with three asynchronous processors, it was often difficult to isolate where any particular problem arose—the 2B ESS, the SPUC/DL, or the application (AMARC, EADAS, or RSS).

Third, debugging tools for the system as a whole lacked some desirable features. The 2B ESS laboratories had powerful debugging hardware for halting the processor, dumping memory locations, tracing execution paths, etc. These facilities were not available on the SPUC/DL, particularly in the late stages of the project when factory-made

units were installed in the laboratory to find any remaining troubles in the production hardware. There were several occasions where it would have been very desirable, for example, to have an event within the ESS halt the microprocessor within the SPUC/DL. We mention in passing that a commercially obtained data communications analyzer proved to be a key instrument in finding certain protocol problems. Ours could record data traffic onto a cartridge tape, stopping after an event of interest; it could also recognize X.25 frame and packet types.

The last few insidious problems in the SPUC/DL turned up in both the hardware and software. The hardware problems were typically sensitivity to "glitches," causing, for example, an erroneous extra DMA cycle. The software problems were usually those of process synchronization and shared resources. The internal architecture of the SPUC/DL, both hardware and software, supports a high degree of concurrency. This increases throughput and minimizes delay, but leads to these internal process synchronization problems. Perhaps more rigorous monitors for shared resources and synchronization semaphores would have paid dividends during the integration phases.

6.2 Particularly successful areas

The problems outlined above led to intense activity as the project neared completion. We were relieved, however, to experience a happy ending; the underlying problems were found and corrected for all observed symptoms.

The climax of our development effort came with the acceptance test for the first 2B ESS—10A RSS field installation. During the 24-hour test, some 60,000 telephone calls were originated on the RSS, processed by the 2B ESS through the SPUC/DL, and properly completed back to the RSS. There were no indications of any problems in the data-link area.

It seems to us that one factor contributing to this success was the attitude that the problems should be *understood* and solved. It might be possible, for example, to stop printing an error message if the message announces some event that seems not to have a harmful effect on the system. We have chosen to explore such events as deeply as resources permit, with gratifying results.

Thus, we feel we have a data-link system essentially free from operational errors. It remains to be seen if our reliability goals will be met.

VII. ACKNOWLEDGMENTS

We acknowledge the considerable contributions of several people to various phases of the project. G. Dobrowski was involved with the

preliminary analysis leading to the architecture of the system. Most of the original SPUC/DL hardware design is by T. Peterson. T. Brinkman assisted with the initial versions of the SPUC/DL microprocessor code. W. McCalla was responsible for most of the original design and strategy decisions in the data-link maintenance area. Much of the data-link maintenance coding was done by R. Atkins, E. Bily, and H. Williams. J. Kent and D. Sikora produced the program for the 2B ESS to diagnose the SPUC/DL. We thank D. K. Ford and S. C. Yuan for their contributions to troubleshooting the hardware, G. Mannon for assistance with modem issues, and D. R. Bierma for continuous attention to testing and administering new issues of both hardware and software.

REFERENCES

1. M. S. Sloman, "X.25 Explained," Computer Commun., *1*, No. 6 (December 1978), pp. 310–27.
2. D. W. Davies, D. L. A. Barber, W. L. Price, and C. M. Solomonides, *Computer Networks and Their Protocols*, New York: John Wiley and Sons, 1979.
3. American Telephone and Telegraph Company, Bell System Technical Reference "Operations Systems Network Protocol Specification: BX.25, Issue 3," Publication 54001, June 1982.
4. Special issue on No. 10A Remote Switching System, B.S.T.J., *61*, No. 4 (April 1982).

AUTHORS

C. E. Ishman, Argonne National Laboratory 1961–1967; Bell Laboratories, 1967—. At Argonne National Laboratory, Mr. Ishman worked on hardware for the zero-gradient synchrotron. Since joining Bell Laboratories he has contributed to many system- and maintenance-software areas on the No. 2 ESS, including call-processing features, both diagnostics and fault-recovery for peripheral units, and conversion of the generic program to the 3ACC processor for the 2B ESS. He is presently a member of the No. 5 ESS System Test Department.

Richard B. Sanderson, B.S. (Electrical Engineering), 1971, Northwestern University; M.S. (Electrical Engineering), 1972, Stanford University; Bell Laboratories, 1972—. Mr. Sanderson has worked on peripheral circuit design for the number 1, 1A, and 2B ESS machines. He is presently Supervisor of the International Circuit Design Group. Member, Tau Beta Pi, Eta Kappa Nu, IEEE.

Louis M. Taff, S.B. (Physics), 1963, Massachusetts Institute of Technology; Ph.D. (Nuclear Physics), 1969, Iowa State University; University of Groningen, Netherlands, 1969–1976; Fermi National Accelerator Laboratory, 1976–1980; Bell Laboratories, 1980—. At the University of Groningen, Mr. Taff engaged in nuclear physics research and became extensively involved with real-time data acquisition systems. At Fermilab, he was a principal in the production of a packaged software system for data acquisition and analysis, now in wide use in high-energy physics. The data-link portion of the 2BE3 generic program was his first project at Bell Laboratories. Mr. Taff is currently working on performance-measurement instrumentation for the number 1A ESS. Member ACM, APS, Sigma Xi.

Donald P. Truax, B.S. (Electrical Engineering), 1966, Michigan State University; M.S.E. (Computer Technology), 1967, University of Michigan; Bell Laboratories, 1966—. Mr. Truax has been involved in the development of fault-recovery, diagnostic, and system integrity programs for No. 1 ESS. Projects have included EADAS and ACD data-link maintenance, and HILO and EPSCS trunk maintenance. Mr. Truax also worked on the PUC and PUC-DL maintenance for No. 1 ESS. He has supervised maintenance projects for both No. 1 ESS and No. 2 ESS. No. 1 ESS projects included PUC-DL maintenance for RSS, ETS, and CCIS and diagnostic improvements for the Remreed networks. Mr. Truax currently supervises the MTSO Maintenance Group for the Advanced Mobile Phone Service project. Member, IEEE, Eta Kappa Nu, Phi Kappa Phi.

Charles T. Tulloss, B.S.E.E., 1974, Tennessee State University; M.S.E.E., 1975, Ohio State University; Bell Laboratories, 1974—. Prior to his work on the 2BE3 data link, Mr. Tulloss has worked in the area of processor-to-processor communications and protocols on the Enhanced Private Switched Communication Service (EPSCS) and the Electronic Tandem Service (ETS) projects. Currently Mr. Tulloss is involved in hardware test and evaluation for No. 5 ESS. Member, Eta Kappa Nu, IEEE Computer Society.