AT&T Bell Laboratories Technical Journal
Vol. 63, No. 6, July–August 1984
Printed in U.S.A.

AT&T Technologies Implementation of Local Area Data Transport—A Hardware and Software Overview

By D. J. STELTE,* H. J. KAFKA*, and W. J. PAULE*

(Manuscript received September 12, 1983)

AT&T Technologies has implemented hardware and software components that will provide an economical Local Area Data Transport (LADT) service. The AT&T Technologies LADT Generic 1.0 is composed of a No. 1 PSS packet switch, one or more statistical multiplexers called Data Subscriber Interfaces (DSIs), and an Administrative Processor (AP) responsible for the administrative functions of the network. This paper describes AT&T Technologies LADT Generic 1.0 hardware and software architectures of the DSI and AP components of the network.

I. INTRODUCTION

The AT&T Local Area Data Transport (LADT)† system is a packet-switched network that provides local exchange areas an economical data communications capability.[1] It incorporates inexpensive access mechanisms, standard interfaces, high availability, and the potential for quick and ubiquitous deployment.[2]

This system consists of three types of nodes (Fig. 1):

1. Data Subscriber Interfaces (DSIs), a statistical multiplexer that terminates up to 124 customer lines and one high-speed network access link.

2. The No. 1 PSS packet switch, a high-reliability, high-availability

* AT&T Bell Laboratories.

† Acronyms are defined in the Glossary at the end of this paper.

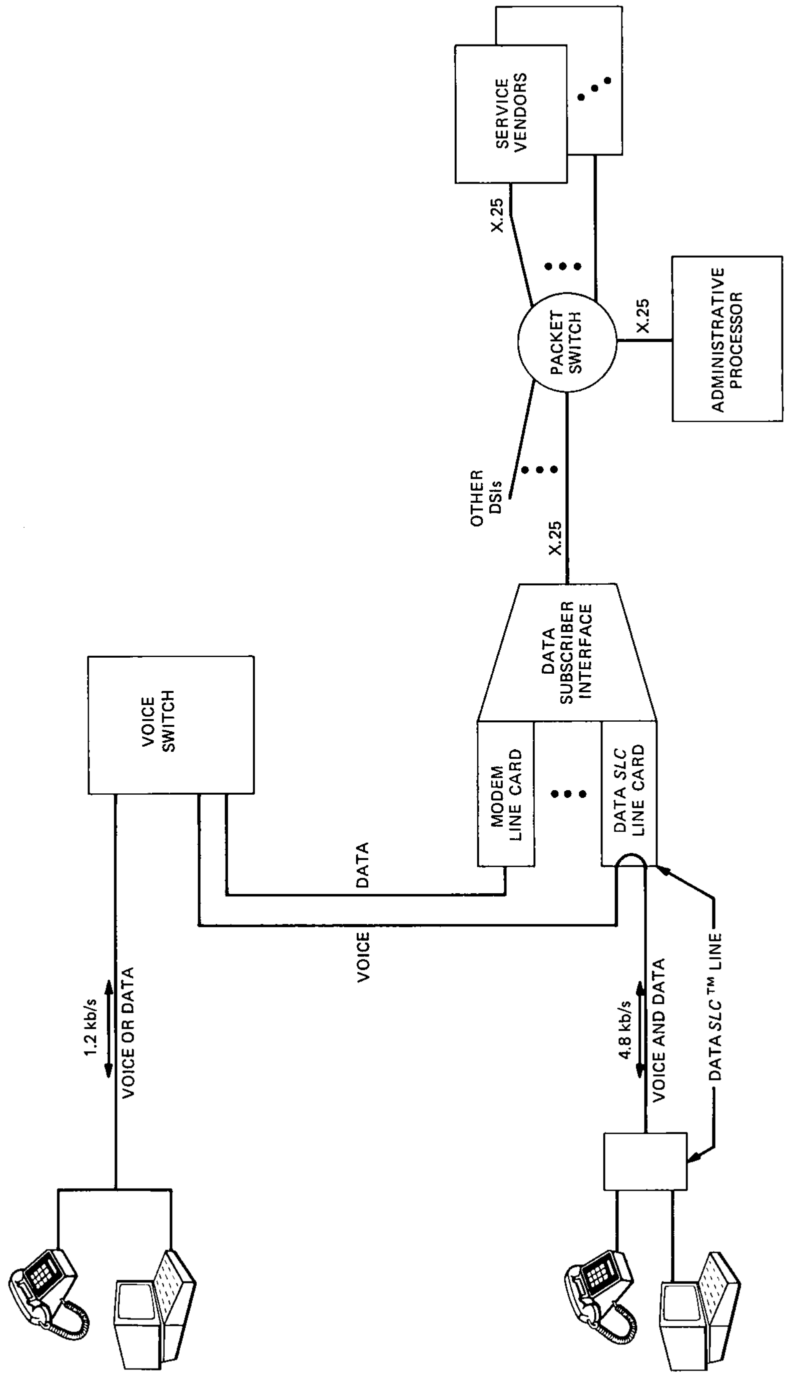

Fig. 1—LADT architecture.

X.25[3] data switch used for routing and transporting packets between DSIs and service vendors.

3. The Administrative Processor (AP), a centralized point for providing maintenance, billing, traffic, craft interface, and network-management functions for the DSIs.

The newly developed DSI provides access and multiplexing functions for subscriber lines. The DSI supports two methods of access for the customer interface. One is a switched access for either voice or 1.2-kb/s data communication through the public switched telephone network. The other is a dedicated access at 4.8 kb/s that provides simultaneous voice and data communication over a single wire pair. The DSI supports Link Access Procedure B (LAPB) as the link-level protocol for subscriber lines.

The No. 1 PSS packet switch is the hub of the network and is used for switching and transporting packets. It connects 56-kb/s lines from DSIs and 9.6- or 56-kb/s lines from service providers. The packet switch provides all essential virtual call services of the 1980 International Telegraph and Telephone Consultative Committee (CCITT) Recommendation X.25 protocol.[3]

Like the packet switch, the Administrative Processor is implemented on a high-reliability, high-availability processor. It provides network support functions in a central location for LADT.

While the preceding article provides an overview of the generic LADT services, this article gives details of the AT&T Technologies implementation of LADT.

II. DATA SUBSCRIBER INTERFACE HARDWARE

The DSI is a statistical multiplexer that concentrates data packets from up to 124 subscriber lines onto a single 56-kb/s data link to the packet switch.[4] The primary purpose of the DSI is to reduce LADT subscriber access costs by sharing packet-switch access costs among many subscribers and by providing inexpensive access to the DSI. To keep costs low while providing high availability, the DSI is implemented as a simplex system that is reliable and easily maintainable.

The DSI is divided into three major subunits, which consist of the processor-complex subunit and two line-group subunits (Fig. 2). The processor-complex subunit contains the intelligence of the DSI, the protocol-handling functions, the network interface, and the craft interface. The processor-complex centralizes the processing power of the DSI so that these functions can be shared over all of the subscriber lines. This reduces the cost by minimizing the individual line interface functions performed in the line-group subunits. Each of the line-group subunits terminates up to 64 lines and multiplexes the data streams from these lines into the processor complex. In line-group subunit 0,

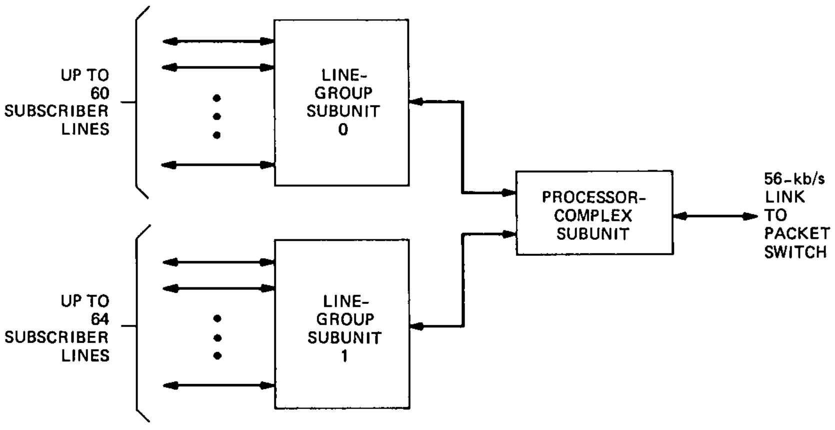
UP TO 60 SUBSCRIBER LINES
LINE-GROUP SUBUNIT 0
UP TO 64 SUBSCRIBER LINES
LINE-GROUP SUBUNIT 1
PROCESSOR-COMPLEX SUBUNIT
56-kb/s LINK TO PACKET SWITCH

Fig. 2—Hardware subunits of a DSI.

4 of the lines are used for test circuitry, while in line-group subunit 1 all 64 lines are available for serving subscribers. The following sections give physical and functional descriptions of the DSI and its components.

2.1 Physical description

LADT subscriber access costs are reduced by minimizing the amount of transmission over lines that are not concentrated. This is accomplished by locating DSIs in the same central offices with voice switches. The DSI hardware is optimized for installation and operation in the central office environment.

The DSI frame, shown in Fig. 3, uses a standard set of devices, apparatus, equipment, and design tools that are common to many AT&T Technologies products.[5] The DSI frame uses a standard framework, which is 7 feet high, 2 feet and 2 inches wide, and 18 inches deep. Each frame contains up to two DSI units and one −48V fuse-panel unit. The base of each frame also contains two −48V filter circuits, one for each of the two power buses entering the frame.

A DSI unit, shown in Fig. 4, occupies 24 inches of vertical space in the frame. Each unit consists of three 8-inch shelves equipped with backplanes. The upper shelf contains the line-group 0 subunit, the middle shelf contains the processor-complex subunit, and the bottom shelf contains the line-group 1 subunit. Each shelf contains its own power converters for the circuit packs in that level so that a power failure in one of the line-group subunits will not result in the failure of the entire unit. A fully equipped DSI unit contains 43 circuit packs.

Four types of cables connect the DSI to other equipment in a central

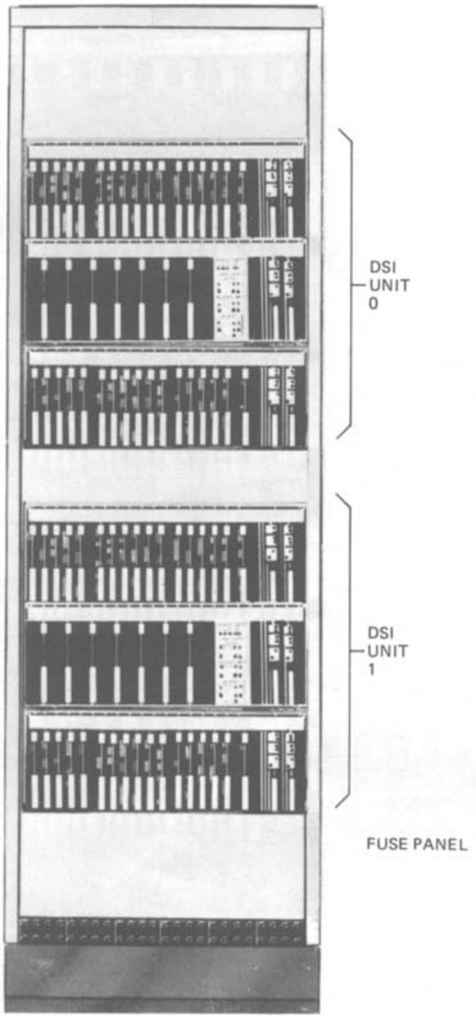

Fig. 3—DSI frame configuration with two DSI units.

office. These cables are installed when the DSI is installed. The cables include:

1. Power cables, which deliver −48V power to the frame from the central office supply.
2. Tip and ring cables, which connect the subscriber-line interfaces on the DSI to the main distributing frame in the central office.
3. 56-kb/s link cables, which provide the interface between the DSI and the packet switch.
4. Central office cables, which connect to alarm and scan points of the central office through the main distributing frame.

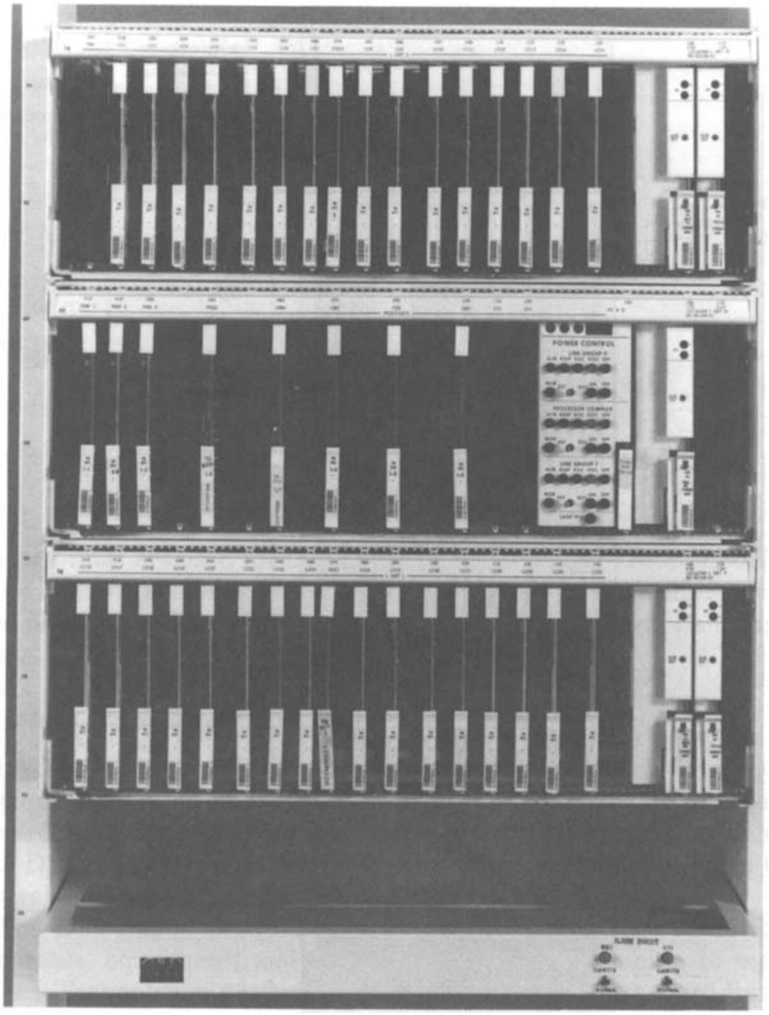

Fig. 4—DSI unit.

2.2 Processor-complex subunit

The processor-complex subunit is designed to provide all of the protocol handling and processing power required in the DSI. The tasks that must be performed are functionally partitioned between specialized hardware components and two general-purpose microprocessors in order to provide optimal performance and flexible service capabilities at a low cost.

Figure 5 shows the functional components of the DSI processor complex. The main-processor circuit pack contains a general-purpose

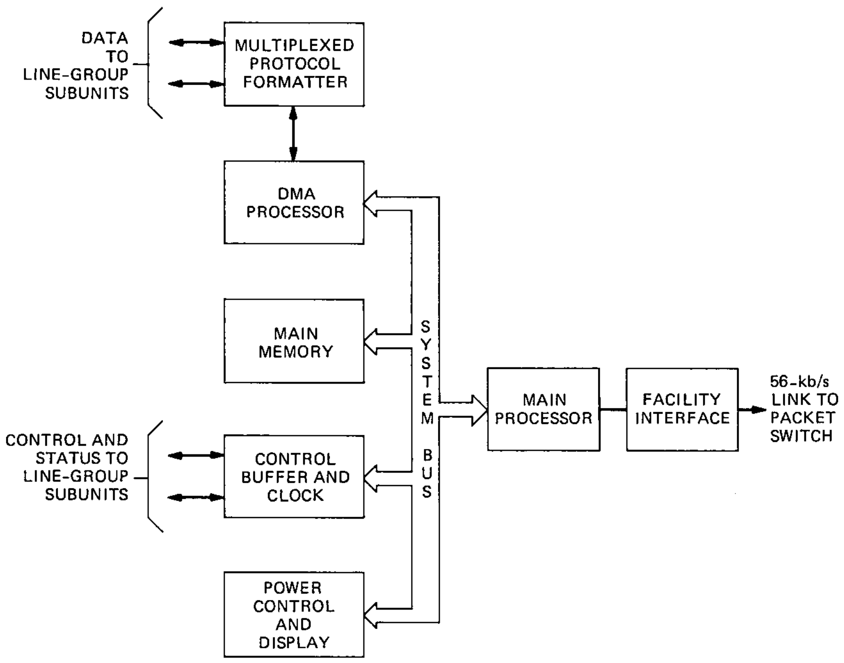
DATA TO LINE-GROUP SUBUNITS
MULTIPLEXED PROTOCOL FORMATTER
DMA PROCESSOR
MAIN MEMORY
SYSTEM BUS
CONTROL AND STATUS TO LINE-GROUP SUBUNITS
CONTROL BUFFER AND CLOCK
POWER CONTROL AND DISPLAY
MAIN PROCESSOR
FACILITY INTERFACE
56-kb/s LINK TO PACKET SWITCH

Fig. 5—DSI processor-complex subunit.

processor that handles call processing, the higher levels of protocols, and general control functions. The main-memory circuit packs provide memory for the main processor's programs and for data buffers. The power-control-and-display circuit pack controls all of the power converters in the DSI and provides the local craft interface for the DSI. The facility-interface circuit pack provides electrical interfaces for the 56-kb/s data link to the packet switch. The control-buffer-and-clock circuit pack generates clocks for the DSI and provides the main processor with convenient access to the line-group subunits. The Direct Memory Access (DMA)-processor circuit pack converts between buffered subscriber data packets and data bytes, and the multiplexed-protocol-formatter circuit pack converts between data bytes and data bits as it handles the lower levels of the protocol on the subscriber lines. The following sections provide more details on these seven types of circuit packs in the processor-complex subunit and on the system bus that interconnects them.

2.2.1 System bus

As Fig. 5 illustrates, the processor complex is organized around a general-purpose microprocessor bus called the system bus. Most of the communication between various components of the processor-complex subunit occurs over this bus. The system bus consists of a 20-bit address bus and a 16-bit data bus. The data and address signals are both protected by parity bits. Other signals that are considered to be a part of the bus include read and write strobes, circuit-pack selects, interrupt requests, bus arbitration signals, and bus timing signals.

2.2.2 Main processor

The main-processor circuit pack contains the controlling intelligence of the DSI in the form of a microprocessor. This microprocessor is responsible for the higher levels of protocols, maintenance, and line control commands, and in general the rest of the DSI. As shown in Fig. 6, the main-processor circuit pack also contains general-purpose microprocessor support circuitry, such as bootstrap Read-Only Memory (ROM), scratch-pad Random Access Memory (RAM), bus controls, interrupt controls, direct-memory-access controls, memory and I/O controls, a sanity timer, and an AT&T Technologies X.25 Protocol Controller[6] (XPC) integrated circuit.

The main-processor circuit pack is based on an Intel 8086 microprocessor operating at 5 MHz, with associated clock and reset circuitry. The main-processor circuit pack also includes 64K bytes of bootstrap ROM and 4K bytes of static RAM that are used during system initializations. The bootstrap ROM stores initialization programs, diagnostic programs, and a download program. These programs enable the DSI to obtain its operational software by placing an X.25 call to the AP through the packet switch. The RAM serves as a temporary scratch-pad memory that is used until the main processor can verify the integrity of the main-memory circuit packs. After the main-memory diagnostics are completed, the RAM on the main-processor circuit pack is disabled and logically replaced by a section of the main memory.

Large amounts of customer data pass between various components of the processor-complex subunit and the data buffers in main memory. To make these data transfers as efficient as possible, the main processor permits other components of the DSI to access the main memory directly by granting them control of the system bus. The main processor includes circuitry to control these direct-memory-access transfers and to arbitrate control of the system bus.

In any system with the hardware and software complexity of the DSI, major faults or illegal states may occur that prevent normal recovery actions from taking place. To facilitate rapid recovery in these situations, the main-processor circuit pack contains a hardware sanity timer. After this timer is initialized, it must be periodically reset by the main processor to keep from expiring. If the main processor

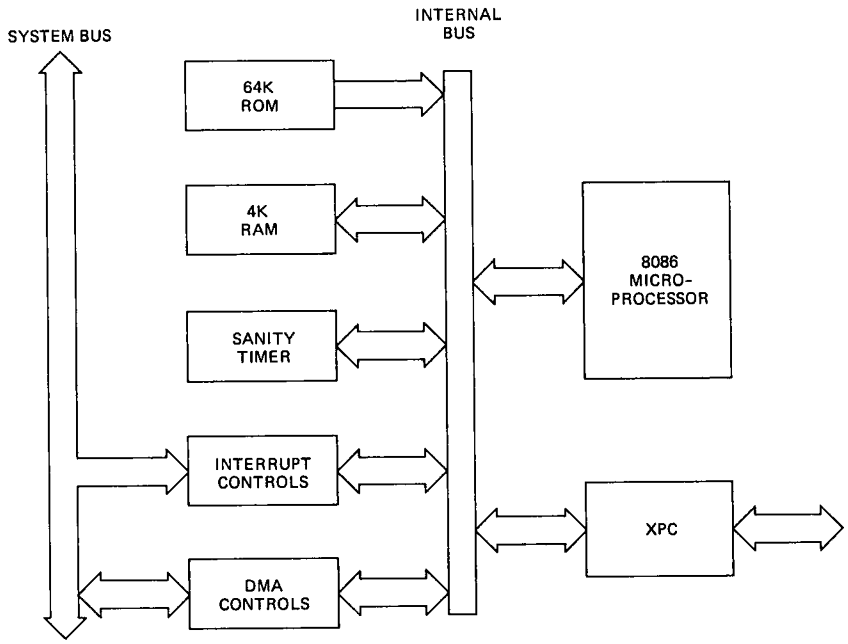

Fig. 6—DSI main processor.

allows the sanity timer to expire, the main-processor circuit is reset by an unmaskable interrupt so that appropriate recovery action can be taken.

The main-processor circuit contains an XPC integrated circuit,[6] which autonomously handles the complete X.25 level-2 (link level) protocol for the 56-kb/s link to the packet switch. The XPC chip handles communications with the packet switch by transferring packets in and out of main-memory data buffers through direct-memory access. The XPC notifies the microprocessor when significant events on the link occur, such as when packets are successfully received or when the packet switch acknowledges the reception of a packet. By autonomously handling the lower levels of the X.25 protocol, the XPC chip enables the microprocessor to concentrate on higher-level functions.

Unusual or significant system events are reported to the main processor by system interrupts. The main processor contains 15 independently maskable interrupt controls with programmable priorities.

2.2.3 *Main memory*

The processor complex has three identical circuit packs that form the DSI main memory. Each of these circuit packs has 256K bytes of

RAM. To increase the reliability of the DSI, the main-memory circuit packs include additional circuitry for write protection and for the detection and correction of both hard and soft memory errors. The DSI main memory provides storage for main-processor operational software, main-processor data, and packet-data buffers.

The memory array on each circuit pack consists of 128,000 words of 22 bits each. Sixteen of the bits store the normal memory word, and the six additional bits provide error detection and correction using a modified Hamming code. The memory array uses AT&T Technologies 64K-bit dynamic RAM chips. Each circuit pack also contains circuitry to refresh the dynamic RAM chips and to arbitrate between memory access cycles and refresh cycles.

A modified Hamming error detection and correction code is used to correct all single-bit memory errors and to detect all double-bit errors. Any errors also cause the generation of an interrupt to the main processor. The Hamming circuitry supports both byte and word reads and writes.

The main-memory write-protection circuitry allows the main processor to write-protect any section's main memory in 512-word (1K-byte) blocks. When an attempt to write into a protected area occurs, the main memory prevents a write from occurring and causes an interrupt to the main processor. Write-protecting sections of memory that contain the main-processor program text help to ensure the integrity of the DSI. When either Hamming or write-protect errors occur, an error register traps the address where the error occurred so that the main processor can take appropriate corrective action.

2.2.4 Power control and display

The power-control-and-display circuit pack in the processor complex contains the local craft interface and the relay contacts for the central office alarm interface. All of the power converters in the DSI are controlled from this circuit pack through the use of switches and Light-Emitting Diodes (LEDs), and other LED displays are used for maintenance and debugging purposes. All of the switches and LED displays are mounted on the faceplate to facilitate access by craft personnel.

The faceplate, shown in Fig. 7, is divided into four areas of indicators and switches. These areas correspond to the three subunits (line group 0, line group 1, and the processor complex) and the craft diagnostic display section. The three subunit areas give the craft control of the DSI and its subunits. By observing the indicator lights and by operating the appropriate switches, the craft can take a subunit out of service, remove power from a subunit, restore power to a subunit, and request restoration to service of the subunit. The lights will also reflect

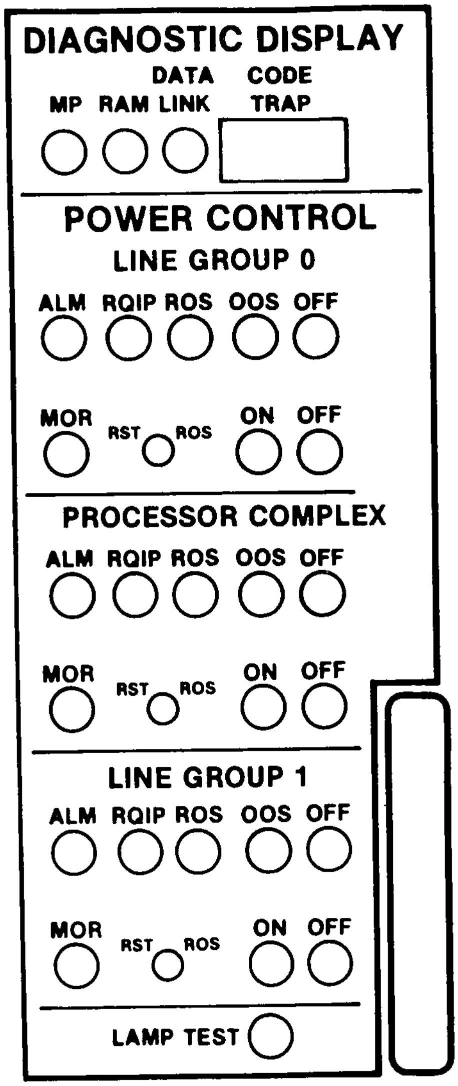

Fig. 7—Faceplate of the power-control-and-display circuit pack.

the status of each subunit when remove/restore service requests are initiated remotely by the AP.

The diagnostic display section of the faceplate contains three LED lights and two 7-segment displays. This diagnostic display section is used to provide fault localization when the link between the DSI and the AP is not functional. During initialization of the DSI, the diagnostic display indicates the correct operation of the DSI main processor, the DSI main memory, and the data link to the packet switch as determined by the ROM-based diagnostics in the main processor. Thus, if a failure occurs during initialization, the craft can make circuit-pack replacements based on the state of the diagnostic display.

2.2.5 Facility interface

The facility-interface circuit pack provides for several electrical interfaces at the 56-kb/s link between the DSI and the packet switch. The DSI can be located either remotely from the packet switch or in the same building as the packet switch. The facility-interface card permits the choice of a transmission mechanism that is cost-effective for each installation of a DSI.

For the case when both the DSI and the packet switch are not collocated, the physical link can be provided by a Digital Data System[7] private digital line in which the DSI looks electrically like Data Terminal Equipment (DTE) and connects to a Data-Service Unit.[8] This general-purpose interface permits the DSI to use standard data transmission equipment to communicate with the packet switch.

For the case when the DSI and packet switch are located in the same building, a direct-link option is provided to reduce the need for data communication equipment external to the DSI. This interface allows the packet switch and the DSI to be directly connected when they are located close together. For slightly longer distances, limited distance modems can be used.

The above two DSI interfaces are supplied as standard equipment and can support any DSI placement currently envisioned. However, since many DSI installations will be in central offices with digital facilities, the facility-interface circuit pack provides an interface that can bypass a substantial amount of the standard customer-interface hardware. This interface, called the DS-O interface because it connects to the digital facilities at the DS-O level,[9] can be used to alleviate the need for both the ac-powered data service unit and the office channel unit. The DSI's ability to directly meet the DS-O interface reduces the overall cost of the packet-network interface.

2.2.6 Control buffer and clock

The control-buffer-and-clock circuit pack contains a clock section, which transmits clock and synchronization signals to many of the other circuit packs in the DSI, and a control-buffer section, which provides a convenient means for the main processor to access registers in the line-group subunits that control and reflect the state of the subscriber lines.

The clock section includes a crystal-controlled phase-locked-loop oscillator, divide chains, and buffers, which provide most of the clocks used throughout the DSI. The crystal-controlled oscillator can free-run or it can be phase-locked to one of several external sources. This synchronization capability permits the subscriber-line data rates to be locked to network clocks. When the DSI is configured to be phase-locked to an external clock source, the control-buffer-and-clock circuit pack reports an error condition whenever the phase lock is lost.

The control-buffer-and-clock circuit pack also contains a control buffer section with a buffer memory and a scanning mechanism. This gives the main processor a simple memory-like interface to registers on the line-interface circuit packs (line cards) that terminate the subscriber lines, without the reliability problems that would result from extending the system bus to all of these circuit packs. This interface permits the main processor to carry out line control and maintenance functions on each subscriber line. The control buffer and clock controls the distribution of the line control information from the control buffer memory to the line cards, and controls the return of status information from the line cards to the status buffer memory. The main processor accesses the control and status information simply by writing to the control memory and reading from the status memory.

2.2.7 DMA processor

The Direct-Memory-Access (DMA)-processor circuit pack and the multiplexed-protocol-formatter circuit pack together handle low-level protocol functions for each of the 128 lines (124 subscriber lines plus 4 test lines) supported by a DSI. The multiplexed protocol formatter (described below) converts individual data bits from the subscriber lines to bytes of information frames. The DMA processor transfers these bytes into and out of main-memory packet-data buffers and controls the multiplexed protocol formatter.

The DMA-processor circuit pack contains an Intel 8086 microprocessor with specialized hardware and ROM-based firmware to put data bytes from the subscriber lines into the DSI main memory, and to take data bytes from the DSI main memory for transmission to the subscriber lines. The DMA processor contains bus interface hardware that enables it to transfer data bytes in and out of main memory with direct-memory-access techniques. The DMA processor also contains hardware for direct communication with the main processor. This hardware includes First-In First-Out (FIFO) memories for commands and responses, a shared memory for line states and packet-buffer addresses, and special registers. Other hardware on the DMA-processor circuit pack includes local static RAM, and specialized hardware to interface to the multiplexed protocol formatter. The DMA processor participates in certain maintenance and line control functions because it acts as the interface between the main processor and the multiplexed protocol formatter. The firmware that controls the DMA processor is described later in this paper.

2.2.8 Multiplexed protocol formatter

The multiplexed-protocol-formatter circuit pack is a time-shared state machine that uses specialized hardware and memory to handle frame-sublevel protocol functions, including flag and frame recognition, bit stuffing and unstuffing, byte assembly and disassembly, and Cyclic Redundancy Check (CRC) generation and checking, for up to 128 lines. The multiplexed protocol formatter handles all 128 lines on time-shared hardware. This reduces costs and simplifies maintenance by using less hardware and by eliminating the complexity of having 128 independent protocol-handling devices. The multiplexed protocol formatter does not interface directly to the system bus. Instead, it is directly controlled by the DMA processor (and therefore indirectly controlled by the main processor). The multiplexed protocol formatter also interfaces to each line-group subunit through time-multiplexed data streams, which contain data bits for all of the subscriber lines in the line-group subunits.

To provide the above functions, the multiplexed protocol formatter consists of two finite-state machines that are time-shared among all customer channels. The basic structure of the multiplexed protocol formatter is shown in Fig. 8. One state machine performs transmission functions, and the other state machine performs reception functions. Each of these state machines is configured for a particular channel immediately before action is required to service that channel. This configuration, or state information, is stored for each channel's receiver and transmitter in the state memory sections shown in Fig. 8. After the channel has been serviced, the new state is stored in memory and the present state of the next line is obtained. By performing these operations at a rate equal to the combined line rates of all channels, the multiplexed protocol formatter handles its portion of the protocol for all lines.

The multiplexed protocol formatter also contains command memories and FIFOs that interface with the DMA processor. The DMA processor passes information to the multiplexed protocol formatter by writing into the command memories. The information in these memories includes line control information and data bytes to be transmitted to the subscriber lines. These memories have separate locations for each line, so the DMA processor can effectively control all of the lines simultaneously. The multiplexed protocol formatter passes information to the DMA processor by writing into FIFO memories. The information includes line-status information and data bytes received

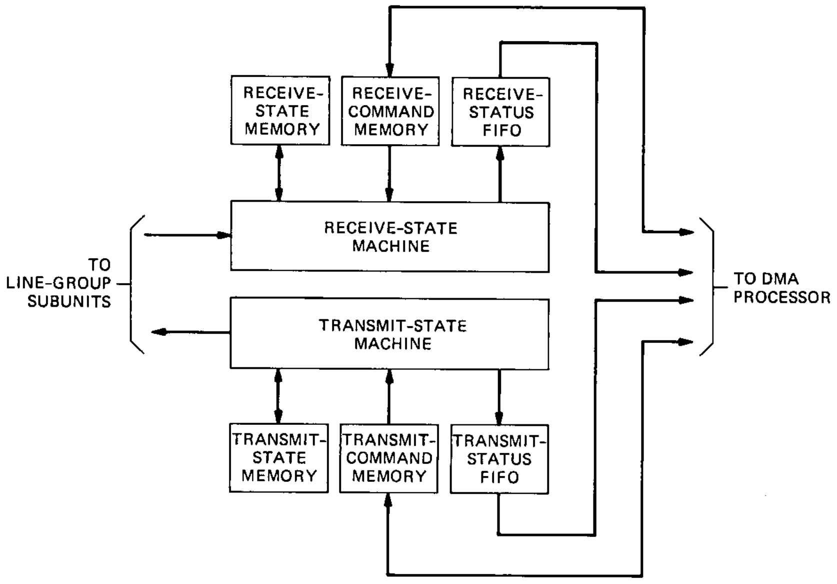

Fig. 8—DSI multiplexed protocol formatter.

from the customer lines. Each entry also includes a line number, which tells the DMA processor the subscriber line to which the entry applies.

2.3 Line-group subunit

Figure 9 is a functional representation of a DSI line-group subunit. The line group consists of one group-distributor-circuit circuit pack and up to 16 line-interface circuit packs, or line cards. The line-group subunits of the DSI may be configured to support various mixtures of line interfaces because the line cards that terminate subscriber lines meet the same backplane interface. The mixture of subscriber lines that terminate on a DSI is determined by the types of line cards that appear in the line-group subunits.

The line cards contain the interfaces to the subscriber lines. Each subscriber line terminates on one of the line circuits on a line card. These line circuits convert the modulated or multiplexed data format on the subscriber line to a digital data stream. Each line circuit also responds to control bits and supplies status bits that are used for line control and maintenance functions. Each line card contains three or four line circuits and a line-card common circuit, which converts the data, control, and status information into the proper format for the backplane interface to the group-distributor circuit. The initial types of line cards for the DSI include modem line cards with four modem

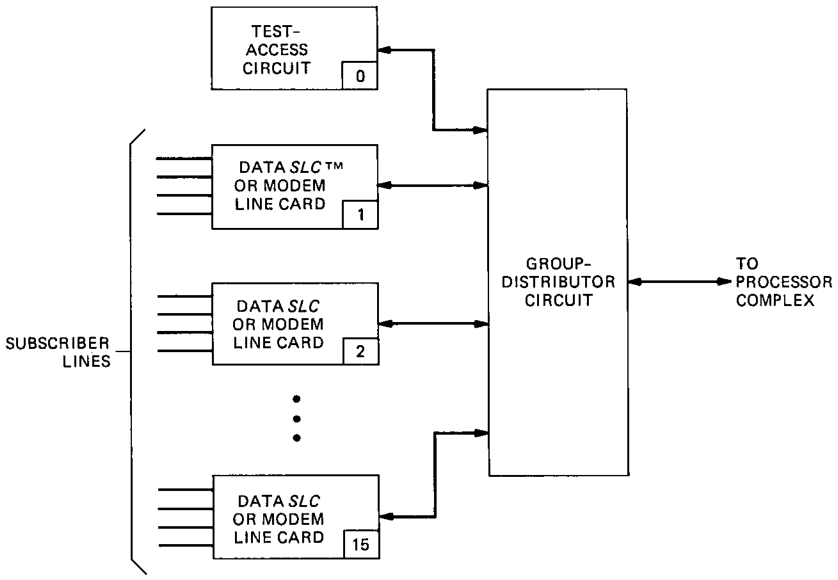
TEST-ACCESS CIRCUIT
0
DATA SLC™ OR MODEM LINE CARD
1
DATA SLC OR MODEM LINE CARD
2
DATA SLC OR MODEM LINE CARD
15
SUBSCRIBER LINES
GROUP-DISTRIBUTOR CIRCUIT
TO PROCESSOR COMPLEX

Fig. 9—DSI line-group subunit.

line circuits per circuit pack, and data *SLC** (Subscriber Line Concentrators) line cards with three data *SLC* line circuits per circuit pack.

2.3.1 Group-distributor circuit

The group-distributor circuit pack distributes clocks, data information, and control information from the processor complex to the individual line cards. Since the multiplexed protocol formatter requires the data bits from the line cards to be time-multiplexed, the group-distributor circuit performs the necessary multiplexing/demultiplexing between the information format for the processor complex and the information format for the line cards. The group-distributor circuit gives each line card a separate set of signals. This permits all of the line cards to have the same interface to the group-distributor circuit. It also increases the maintainability and reliability of the line group by isolating the effects of line-card failures. The group-distributor circuit also contains circuitry to assist in the maintenance of the control information paths to the line cards.

2.3.2 Modem line card

The modem line card is currently expected to be the most common

* Trademark of AT&T Technologies, Inc.

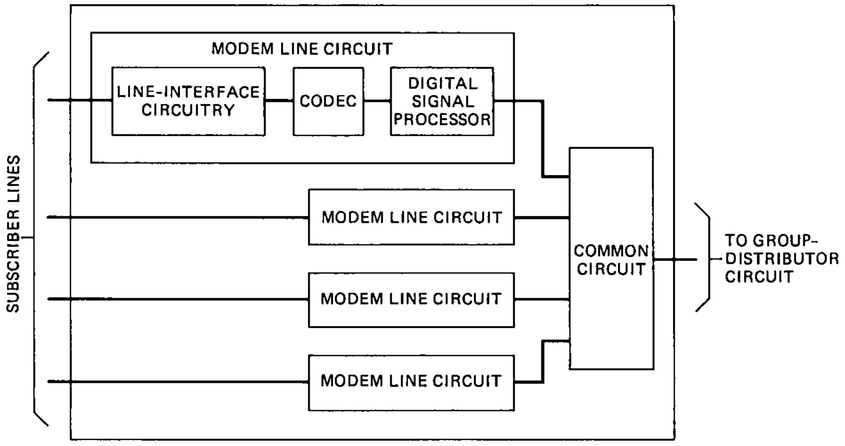

Fig. 10—DSI modem line card.

of the DSI line cards. The modem line card has four line circuits, each of which interfaces to one subscriber line. As Fig. 10 indicates, the modem line card consists of a common circuit, which is shared over all of the subscriber lines, and four modem line circuits (one for each subscriber line). Each modem line circuit performs the functions of an answer-only, 1.2-kb/s 212A modem[10] operating in the synchronous mode. The common circuit includes clock circuitry for the individual modem line circuits and interface circuitry for meeting the common backplane interface to the group-distributor circuit. This common circuit is implemented in a custom Metal Oxide Semiconductor (MOS) device.

Each of the modem circuits on a modem line card includes a Digital Signal Processor (DSP) integrated circuit[11] made by AT&T Technologies, which does the actual modulation and demodulation; a codec, which converts between the analog signals on the subscriber line and the digital format used by the DSP; and circuitry to interface to the subscriber line. The DSP performs most of the data functions of the 212A modem, including high-order digital filters for modulation and demodulation, frequency generation for modulation, carrier detection, clock recovery, and data scrambling/descrambling. The line-interface circuitry includes a ringing detector (since the modem line card is alerted to incoming calls by the presence of ringing on the subscriber lines), a 2-wire to 4-wire hybrid, and line-control relays.

Each line-card slot in the line-group subunit backplane is equipped with shorting contacts, which close whenever a line card is removed from the slot. In the case of modem lines, these contacts make the lines appear busy to the central office so that they will be skipped in

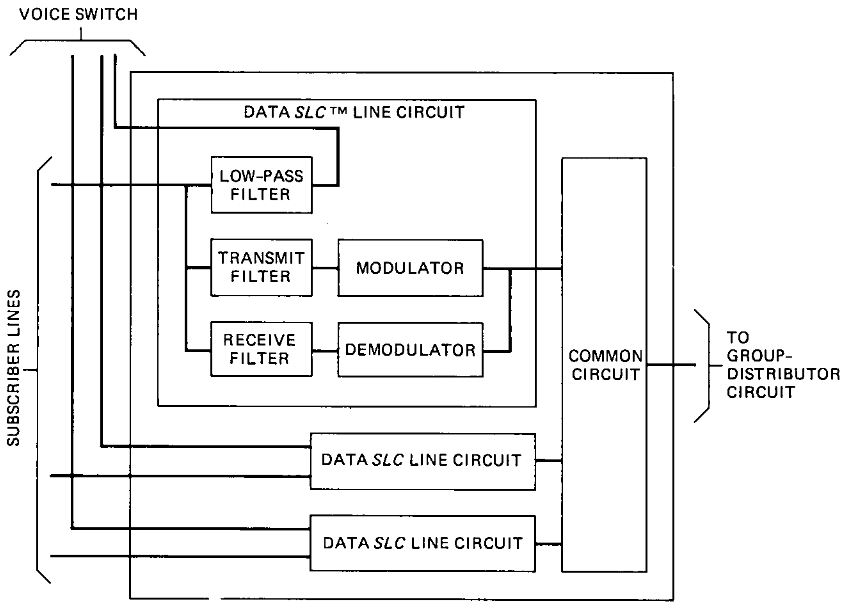

Fig. 11—DSI data *SLC*™ line card.

the line-hunting sequence. This prevents calls from terminating to lines that are temporarily unequipped in the DSI.

2.3.3 Data SLC line card

Data *SLC* system provides simultaneous voice and data channels on a single pair of wires through a technique similar to the *SLC*-1 subscriber carrier system, except that the derived voice channel of the *SLC*-1 system is replaced by a data channel in the data *SLC* system. As Fig. 1 indicates, the data *SLC* system loop is terminated by a data *SLC* line card at the DSI and by a data *SLC* remote terminal at the customer premises. The data *SLC* line card and remote terminal separate the data and voice channels in order to maintain two independent paths: a voice path between the central office and the normal subscriber telephone equipment, and a data path between the subscriber data terminal and the DSI.

As Fig. 11 indicates, the data *SLC* line card consists of a common circuit, which is shared over all three of the subscriber lines, and three line circuits, each of which is dedicated to one subscriber line. The data *SLC* line-card common circuit includes interface circuitry for meeting the common backplane interface to the group-distributor circuit, and clock circuitry that generates the clocks that are needed by the line circuits. The line circuits include passive filters to keep the high-frequency data carrier signals from being sent to the central office voice switch. Since these filters are passive, power failures in the line-group subunit do not affect the integrity of the voice path. Other filters in the line circuit separate the data channels from the voice channel and from each other. The data channel uses Frequency Shift Keying (FSK) modulation with carrier frequencies of 76 kHz and 28 kHz for data transmitted to the subscriber and to the DSI, respectively. The line circuits include circuitry to send a message to the data *SLC* remote terminal in order to invoke "far end loopback", as well as circuitry to perform a digital loopback on the data *SLC* line card itself. These loopback points permit the main processor to verify the integrity of the data paths.

Since the data *SLC* line card and the remote terminal are involved in both the data path and the voice path, they have been carefully designed to ensure the integrity of the voice channel in the event of power failures and failures in the data channel. Occasionally, failures in the data *SLC* line card require its replacement. To avoid disruption of the voice paths while repairs are being made, the line-group subunit backplanes contain shorting contacts. These contacts close whenever the data *SLC* line card is removed from the DSI, connecting the subscriber loop directly to the central office. This ensures the continuity of the voice path between the subscriber and the central office voice switch when the data *SLC* line card is removed from the DSI.

2.3.4 Test-access circuit

Each DSI contains one test-access circuit pack, which is used for diagnosing and maintaining line cards. The test-access circuit facilitates fault isolation by simulating customer premises equipment in order to determine whether a fault on a subscriber line is caused by a line card in the DSI.

The test-access circuit occupies the first line-card position in line-group subunit 0. It contains circuitry that simulates dial-up and direct subscriber lines, including an originate-only modem, a data *SLC* remote terminal, a ringing generator, and loss insertion circuits. Under command of the main processor, any line circuit in the DSI can be temporarily disconnected from a subscriber line and connected to the test-access circuit. This permits the DSI to thoroughly diagnose the line circuit and to localize faults within the DSI.

2.4 Data SLC remote terminal

As Fig. 1 shows, the data *SLC* remote terminal terminates the data *SLC* loop on the customer premises. Although the remote terminal draws ac power from the customer premises, it is part of the network and serves as the Network Channel Terminating Equipment. It uses a passive low-pass filter to pass the voiceband to standard telephones. The modulation and demodulation circuitry is similar to the circuitry of the data *SLC* line circuit. Other circuitry in the data *SLC* remote terminal performs clock recovery and provides the interface to the subscriber data terminal. This is a 4-wire interface using baseband, bipolar, return-to-zero signaling.[12] The data *SLC* remote terminal also contains circuitry to loop back the data path for maintenance purposes.

2.5 DSI hardware provisions for maintenance

The DSI hardware design incorporates many features to detect and isolate hardware faults. These features help to ensure high availability by detecting and isolating minor faults before major service disruptions occur and by localizing faults to circuit packs so that rapid repairs can be made by replacing the circuit pack. The circuitry that implements these features falls into two main classes: operational error detection circuitry and diagnostic error detection circuitry. The operational error detection circuitry continuously and automatically checks for errors while the DSI is operating normally. If an error is found, it is reported to the main processor so that corrective action may be taken. The diagnostic error detection circuitry provides thorough operational checks and fault localization. Since this diagnostic error detection circuitry could interfere with normal DSI operations, it must be explicitly requested by diagnostic software.

2.5.1 Operational error detection circuitry

Several types of error detection circuitry operate automatically and notify the main processor only when errors occur. The parity bits sent over the processor-complex bus fall into this category. During every bus cycle involving a transfer of information between two circuit packs, the circuit pack that is driving the address bus generates the address parity bit, and the circuit pack driving the data bus generates the data parity bits. The circuit pack that reads the bus also checks bus parity. If any error is detected, an interrupt notifies the main processor. This permits the rapid detection of bus faults.

Parity bits are also sent between the control buffer and clock and the line cards in order to verify the integrity of the control and status data path. The method of generating and checking these parity bits allows the control buffer and clock to verify the integrity of this path both to and from the line cards. The control buffer and clock generates a parity bit over the control data, and sends these data through the group-distributor circuit to a line card. The line card determines the sense of the parity bit (odd or even), and then generates a parity bit over the status information that has the same sense (odd or even) as the parity bit that it received. The line card sends the status information and parity bit back to the control buffer and clock through the group-distributor circuit. The control buffer and clock checks the sense of the parity over the status information to confirm that it matches the sense of the parity that it sent out with the control information. If an error is detected, the main processor is alerted.

By varying the sense of the parity in a particular manner, the control buffer and clock uses the parity mechanism to verify other aspects of the control and status scanning procedure. In each cycle of scanning the 128 lines, the control buffer and clock sends out odd parity to 127 of the lines and even parity to the other line. The line that receives the even parity is changed from one cycle to the next, so that all lines receive even parity once in every 128 cycles. With this procedure, the control buffer and clock confirms the integrity of the control and status path, confirms the parity checking and generating hardware on the line cards, and confirms that it is transmitting to and receiving from the proper line card at each point in the cycle.

The multiplexed protocol formatter and the line cards exchange parity over the data bits in a manner similar to the exchange between the control buffer and clock and the line cards. If the multiplexed protocol formatter detects any parity errors, it notifies the DMA processor, and the DMA processor notifies the main processor. In this way, all of the major data paths in the DSI are continuously monitored for integrity through the use of parity bits. This permits the main processor to detect and respond to faults very rapidly.

2.5.2 Diagnostic circuitry

The DSI contains diagnostic error detection circuitry, which enables the main processor to verify the proper operation of the circuitry, to localize any detected faults to a particular component, to verify the repair of the DSI, and to reconfigure the DSI so that it can operate in the presence of certain faults.

The localization of faults is accomplished principally through loopback points. At a loopback point, the main processor can order the transmit data to be looped back to the received data. This enables the main processor to compare the received data to a known pattern that it is transmitting in order to verify that the circuitry up to the loopback point is operating properly. Loopback points occur at strategic locations in the DSI in order to permit the detection of faults and to permit the determination of the location of faults. Since the data coming from the customer are ignored when a loopback is activated, loopback tests are only performed when customers are not being served by the portion of the circuitry under investigation.

The DSI contains diagnostic circuitry, which permits the main processor to inject faults in order to verify that fault detection circuitry is operating properly. As an example, the main processor can force bad parity to be generated by the multiplexed protocol formatter in order to confirm that the bad parity is sent to a line card, returned to the multiplexed protocol formatter, and reported back to the main processor.

III. LADT SOFTWARE OVERVIEW

To make the network more flexible and cost-effective, functions in the LADT system are divided between two different types of processors. The functions that provide the interface to subscriber lines tend to be real-time intensive and are performed in small, microprocessor-based concentrators called Data Subscriber Interfaces (DSIs). The DSIs were kept simple to make them both reliable and cost-effective.

Functions such as billing and traffic-data processing, required to administer an LADT network, were centralized in the AP, the highly reliable duplex computer. Since only one AP is required per LADT network, it can have resources such as terminals, printers, and disk storage that cannot be economically provided on each DSI. The disk storage in the AP is used to store the operational software of the DSIs so that they do not need to have local mass storage. The AP also provides a convenient central location from which the network can be administered.

Many LADT functions are divided between the DSIs and the AP. For example, traffic data for subscriber lines are collected in the DSIs, then sent to the AP for processing into printed reports. Table I shows how functions in LADT are partitioned between these two network nodes. The remainder of this section will discuss some major LADT functions and how they are divided between the DSIs and the AP.

3.1 AP/DSI communication

Since many logical functions are split between the AP and the DSIs, a mechanism was created to allow the two parts of each function to communicate. During normal operation, each DSI always has one X.25 virtual call up to the AP. The Remote Internal Communication Handler (RICH) implements another layer of protocol above the X.25 level 3 to handle the multiplexing of the many logical data streams between the AP and the DSI. It allows individual processes within the AP and DSI to communicate by providing an addressing scheme compatible with the LADT internal architecture and adds another layer of error checking.

3.2 Craft interface

An LADT network can be distributed over an area at least the size of a major metropolitan area. The AP provides a centralized craft

Table I—Division of functions between AP and DSIs

Function	AP	DSI
Subscriber call processing		X
Billing data collection		X
Billing data processing	X	
Traffic data collection		X
Traffic report generation	X	
DSI diagnostics		X
DSI trouble-locating procedures	X	
Craft interface	X	
DSI generic storage	X	
DSI recent change craft interface	X	
Central LADT recent change database	X	
Local copies of DSI database		X

interface for the geographically distributed DSIs, with only a limited subset of the craft interface available at the DSI. Locating the craft interface for the network in the AP lowers the number of craft needed to administer the network, makes the DSIs less expensive, and allows the DSIs to be located in unattended offices.

DSIs and DSI subunits can be restored to service, removed from service, and diagnosed from the AP. All recent changes in data for the AP and the DSIs are entered at the AP.

In addition to the maintenance terminal and recent change terminal, which are standard on the AT&T 3B20D computer, LADT provides remote terminals for interfacing to other telephone operating company work centers such as the network administration center, the recent change and memory administration center, and the switching control center. Each center has a command set tailored to run only the LADT commands needed at that center, with only the switching control center and the local terminals allowed to run the entire set of LADT commands.

3.3 Data call processing

The data-call-processing functions of LADT are performed by the DSIs and the packet switch. The packet switch provides a standard CCITT recommendation X.25 interface for direct 9.6- and 56-kb/s lines, while the DSI provides an X.25 level-2 (LAPB) interface for 4.8-kb/s direct subscriber lines and 1.2-kb/s dial-up subscriber lines. With this configuration, the DSI is treated as a host DTE (Data Terminal Equipment) by the packet switch, while the DSI provides an X.25 LAPB DCE (Data Circuit-Terminating Equipment) interface to the subscriber terminal.

The DSI data-call-processing software is responsible for supporting the X.25 protocol on a single 56-kb/s access line to the packet switch and supporting the LAPB protocol for up to 124 subscriber lines. The DSI establishes an X.25 virtual circuit through the packet switch to a remote X.25 host for each call initiated on a LAPB access line. Once the data call setup has been performed, subscriber data flow transparently on this virtual circuit until either end disconnects the call.

3.4 Billing

Billing data collection for subscribers with direct access is done in the DSIs. Single-entry billing is done for calls that cross less than two midnight boundaries. At the end of one of these "standard" calls, the DSI sends a data-call record to the AP that contains connect and disconnect times, packet counts for up to four different rate periods, and a variety of "per-call" data that may be used for detailed traffic studies. If a call extends past two midnights, at each midnight after the first, the DSI sends an interim record to the AP and clears its internal packet counters. A final record is then sent at the end of the call. This method prevents counter overflow on very long calls and prevents all the data from a long call being lost if a DSI failure occurs. The DSI can buffer several hours of billing records if the link to the AP is temporarily unavailable.

3.5 Maintenance

The LADT maintenance functions are divided between the AP, the packet switch, and the DSIs. Each of these units has local diagnostics to detect and localize hardware faults, along with local audits to verify the integrity of the operational software. The AP provides a unified craft interface for maintenance of the AP and of all of the DSIs that communicate with the AP. While the DSI has its own local diagnostics and audits, the DSI reports the results of its diagnostics and audits to the AP, where they are displayed to the craft. The AP also provides a means for the craft to control the execution of diagnostics in the DSI. In order to permit the craft to localize faults that may prevent a DSI from communicating with the AP, the DSI hardware includes a basic display and craft interface at the unit itself. This permits the evaluation of the results of diagnostics that the DSI executes in order to determine why the link to the packet switch cannot be established.

3.6 DSI software generic download

The DSI is a RAM-based processor. The software generic needed by the DSI is downloaded from the AP to the DSI during DSI start-up. The DSI contains only enough ROM to accommodate start-up diagnostics and the software needed to download the generic. Storing the DSI generic in the AP allows the code in the DSIs to be updated without having to change the ROM in each DSI. The generic download function is the only AP-to-DSI communication that does not use the RICH, since the code for the RICH function is part of the generic to be downloaded. Instead, generic download uses its own layer on top of X.25 level 3, optimized to transfer large amounts of data efficiently and with high reliability.

3.7 Traffic

The DSIs collect extensive traffic measurements, which are sent to the AP every 5 minutes for processing. Any errors occurring at the DSI will be reported to the AP immediately. The AP produces two sets of traffic reports, one for normal DSI traffic measurements, and one for DSI error counts. Both types of reports can be generated for 5-minute, 30-minute, and 24-hour periods. The printing of the 5-minute reports is normally inhibited, but they are printed automatically any time a threshold in a report is exceeded. In addition, the craft can manually request a printing of any of the last six 5-minute reports, any of the last 48 30-minute reports, or the last 24-hour report. The content of each of the reports can be tailored through the LADT recent change system.

IV. DSI SOFTWARE ARCHITECTURE

The DSI software system, which executes on the main processor, is divided between ROM-based firmware, which is responsible for bootstrapping the DSI from power-up or system recoveries, and the RAM-based operational software, which is responsible for normal operations of the DSI. The operational software generic is remotely stored on the AP disk file system and is downloaded into DSI main memory by the firmware.

The firmware is designed as a collection of special-purpose programs that are executed sequentially from a single controlling subroutine. Sequential execution is performed at base level while external events such as timer interrupts and packets received from the packet switch access line are processed at interrupt level. Common library routines are available for setting software timers and communicating with the AP. After start-up or system recoveries, the bootstrap firmware is responsible for placing the hardware into a known state, running a minimal set of diagnostics on essential hardware such as the main processor and main memory, establishing an X.25 virtual circuit to the AP for downloading the operational software, and installing the downloaded software into DSI main memory. Once this has been successfully performed, execution will be passed to a known entry point in the operational software. When the operational software begins execution, the real-time DSI Operating System (DSIOS) is initialized; a master-control process is created, which performs data initialization and synchronizes system-process creation; and execution control is turned over to DSIOS.

The core of the DSI operational software is the data-call-processing system, which contains the X.25 protocol programs and access line interface software. Administrative programs collect and maintain LADT administrative data such as billing and traffic measurements. The DSI software includes resident maintenance programs that minimize the impact of failures on system performance and provide the craftsperson with the information required to locate and repair any troubles quickly. Management of the execution of these programs and of system resources is controlled through DSIOS.

The operational software programs execute as processes under DSIOS. A process can be either a system process or a nonsystem process. System processes are permanent, created during system initialization, and should never terminate unless a system reinitialization is performed. Most administrative and maintenance programs execute as system processes. Nonsystem processes are temporary, created upon demand, and normally terminate when the task they are designed to perform is complete. Transient tasks such as hardware diagnostics and call processing execute as nonsystem processes. Except for hardware interrupts, task execution is strictly nonpreemptive. Processes run until they voluntarily return control of the main processor to DSIOS.

4.1 Operating system

The DSI has a general-purpose real-time operating system (DSIOS) designed to meet the specific needs of the operational software. DSIOS manages processes, memory allocation/deallocation, software timers, and intra-DSI process communication. Access to these resources is controlled through a set of DSIOS primitives that have been designed to provide a powerful and efficient, yet simple, interface. A list of the most frequently used DSIOS primitives and their functions is shown in Table II.

Programs execute as processes under DSIOS. Process creation is controlled by the `OScreate` primitive. A program table defines the characteristics of each system program, such as scheduling priority, stack size, and main entry point. At process creation, a stack and process-control block is allocated to the process. The process-control block is used to store the state of the process when it relinquishes the main processor. A process-ready queue is associated with each process priority level. There are three process-ready queues—high, medium, and low. A program's priority is determined by the needs of the entire system. Critical tasks such as DSI clock synchronization are given highest priority, call processing receives medium priority, and deferrable jobs such as maintenance have lowest priority.

When a process is ready to run, it is appended to its respective

Table II—List of frequently used DSI Operating System (DSIOS) primitives

Primitive	Operation
`OScreate`	Create a new process and schedule it to run.
`OSget1type`	Get a message of a specific type. If no message of desired type is waiting, caller can request either immediate return or suspension until message arrives.
`OSsendmsg`	Send a message to a process. If the receiver is waiting for a message of that type, schedule it to run.
`OSrtimer`	Set a timer using the relative clock. A relative timer is used to schedule a process after a specific delay.
`OSatimer`	Set a timer using the absolute clock. Absolute timers are used to schedule a process at a specific time.
`OSktimer`	Kill a timer. This is used by the call processing software to stop protocol timers.
`OSsuspend`	Take a real-time break. The process is placed on the bottom of its priority queue.
`OSexit`	Exit the current process and remove it from the system.

process-ready queue. The DSIOS process scheduler visits the process-ready queues in a fixed pattern. The high-priority queue is visited most frequently and the low-priority queue least frequently. When a nonempty process-ready queue is found, the first process on the queue is removed, the state of the processor is updated by its process-control block, and processor execution is turned over to the process.

Processes within the DSI communicate by passing messages. The `OSsendmsg` primitive allows any process to send a message to any other process, while the `OSget1type` primitive allows a process to receive messages. A process can request a specific message type or any message type. A request for any message will return the first message waiting for the process. Message reception can be blocking or nonblocking. When a blocking `OSget1type` request is made and the requested message is not immediately available, the process relinquishes the main processor. A nonblocking request returns immediately with an indication of whether or not a message was found.

DSIOS memory management is simplified by partitioning memory into preallocated buffer pools. Initializing the DSIOS creates a linked list of free buffers for each buffer type. Allocation and deallocation routines are provided for each buffer pool. When a buffer is allocated, it is removed from its respective free list, and the buffer is tagged with the process identification number for software auditing purposes. This scheme eliminates memory fragmentation, simplifies software audits, and provides a simple environment for software testing and debugging.

Access routines are provided for starting and stopping software timers. Timers are maintained in a circular queue that is serviced on a periodic basis. The circular queue is composed of head cells that point to a linked list of timers. When a timer is set, DSIOS inserts the time on the linked list associated with the appropriate head cell. When the time expires, it is removed from the circular list and placed on the destination process-control block. When a timer stop request is issued, the timer is removed from the circular list, or if the timer has expired, it is removed from the process-control block. Timer delays can be set relative to the current time of day with the `osrtimer` primitive, or for an absolute time of day with the `osatimer` primitive.

4.2 DSI-system start-up

The DSI-system-start-up subsystem is responsible for initializing the DSI to an in-service state for processing subscriber calls. The system is initialized upon start-up or whenever an unrecoverable error is detected. The software for this subsystem is split between the DSI bootstrap firmware that brings the system up from a power-up, and RAM-based operational software that is responsible for initializing the system to an operational state after the operational software has been downloaded into DSI main memory.

4.2.1 DSI boot sequence

Whenever the DSI starts up or the integrity of the operational software is suspect, the DSI must download a copy of the operational software from the AP disk into the DSI main memory. The start-up firmware is responsible for DSI hardware initialization, running out-of-service hardware diagnostics, setting up an X.25 virtual circuit to the AP for receiving the operational software, and downloading the operational software into the DSI main memory.

A goal of the system start-up firmware is to bring up those components of the hardware that are absolutely necessary to communicate with the AP in order to download the operational software. The DSI peripheral hardware associated with interfacing to subscriber access lines (control buffer and clock, DMA processor, group distributor circuits, and subscriber line interface cards) is disabled. Hardware diagnostics are executed on the main processor, main memory, power-control-and-display (PCD), and facility-interface circuit packs to determine whether hardware faults exist that will prevent the DSI from coming into operation. If diagnostics fail, the system cannot be brought into service. The main processor will be halted and a hardware alarm will be asserted. The PCD panel lights will identify the circuit pack and the diagnostic test that failed.

Once the hardware has been initialized and diagnostics have passed, the DSI attempts to establish an X.25 virtual circuit to the AP. If this is successful, main processor control is turned over to the generic

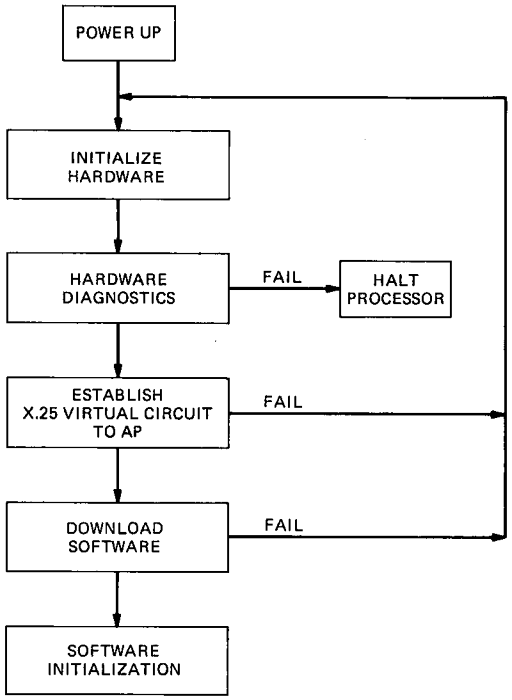

Fig. 12—Flow diagram of DSI boot sequence.

download program, which sends a download request message to the AP. The operational software is transferred from the AP over the X.25 virtual circuit. The generic download program numbers and checksums each message as an added measure of protection. Once the operational software is successfully received, control is passed to a fixed entry point in the operational software. A software initialization will be performed to place the machine in an in-service state for servicing subscriber calls. A flow diagram of a DSI boot sequence is shown in Fig. 12.

4.2.2 Software initialization

Software initialization is the lowest level of DSI recovery. This action is taken immediately after completing a DSI boot or whenever unrecoverable software or hardware faults are detected. A threshold is placed on the number of software initializations allowed during a given period of time. When this threshold is exceeded, a DSI boot is initiated. This causes hardware diagnostics to run and a fresh copy of the operational software to be downloaded.

The entry routine of the operational software initializes DSIOS, creates a process called start-up control, and turns main processor

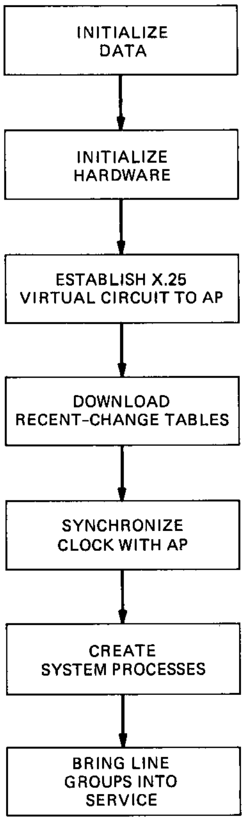
INITIALIZE DATA
INITIALIZE HARDWARE
ESTABLISH X.25 VIRTUAL CIRCUIT TO AP
DOWNLOAD RECENT-CHANGE TABLES
SYNCHRONIZE CLOCK WITH AP
CREATE SYSTEM PROCESSES
BRING LINE GROUPS INTO SERVICE

Fig. 13—DSI start-up flow sequence.

control over to DSIOS. The control flow sequence for the start-up control process, which coordinates bringing the DSI into service, is shown in Fig. 13. The start-up control reinitializes the DSI hardware, initializes shared system tables, and creates all of the system processes. After hardware initialization, the X.25 protocol over the packet-switch access link is reestablished. An AP/DSI communication process is created to establish an X.25 virtual circuit to the AP for receiving critical data and craft requests. Once a communication path is established to the AP, a process is created for downloading the recent change tables from the AP. A process is then created to retrieve the current time-of-day from the AP. The DSI needs an accurate time-of-day clock for call billing purposes and for scheduling time-of-day events. After updating the time-of-day clock, all other system processes are created. The start-up control waits for completion messages from each process before it brings the line groups into service. Once the line groups have been brought into service, the DSI is ready to receive subscriber calls.

4.3 Administrative software

The administrative functions of LADT such as billing, traffic measurements, and recent change are split between the DSI and AP. The DSI administrative software programs serve to collect and update the administrative data. Data are collected and sent to the AP for processing. The processes that perform these operations in the DSI communicate with their counterpart routines in the AP through the Remote Internal Communications Handler (RICH) software subsystem.

4.3.1 DSI/AP communication

Data exchanged between the DSI and the AP are multiplexed over a single X.25 virtual circuit. A DSI process called the RICH process manages this data stream. An access routine sends messages to AP processes. The DSIOS message-sending primitive `ossendmsg` routes messages from the AP to the DSI processes. To simplify the DSI/AP RICH process interface, no flow-control or acknowledgment procedures are used. The features of X.25 level 3 accomplish this. The user process must provide additional data integrity procedures.

The DSI/AP virtual circuit is established during software initialization. The DSI-RICH process attempts to reestablish this connection upon detecting a failure without having the system recovery software intervene. This allows the DSI to provide subscriber service when communication with the AP fails. All transmission requests to the AP are denied when the DSI/AP communication path is not operational.

4.3.2 Billing

The DSI collects billing data for each subscriber call. Data such as holding time and packet counts are maintained in a virtual call record for each call. When a call terminates, the call processing software sends the virtual call record to a DSI per-call billing process. This process then sends the virtual call record to the AP for storage on a magnetic tape.

Because of the importance of the billing information, the design of the per-call billing process prevents the loss of billing data if the communication path to the AP is lost. During an outage, virtual call records are stored in the DSI, and the per-call billing process periodically attempts to retransmit the stored records to the AP.

Calls that extend over two or more consecutive midnight boundaries are given special attention. A special process called long duration billing handles these calls. This process runs every midnight and scans each virtual call record to determine whether it extended over two midnight boundaries. If so, a long duration record is created from the virtual call record and is sent to the AP for storage.

Periodic checks are made to ensure that billing records are not lost between the DSI and the AP. A billing-tracer process sends counts to the AP every hour, allowing the regional accounting office to keep track of billing records. If any billing records are lost, the regional accounting office can detect this by comparing the number of billing records received with the tracer counts.

4.3.3 Traffic

The DSI collects measurements that are used for off-line evaluation of system demand, subscriber calling patterns, service, and DSI component utilization. Counters are pegged by the operational software and are collected and sent to the AP for storage every 5 minutes by a DSI traffic process. Certain measurements, such as queue lengths and the number of active calls, are sampled on a more frequent basis by the DSI traffic process and are also sent to the AP.

4.3.4 Recent change

Certain information in the DSI changes independently of the operational software and is referred to as recent-change data. The recent-change subsystem maintains these data in system tables and provides access routines for other programs. Because of the critical importance of these data, checksums are included as part of the data tables, and the tables are write-protected.

Recent-change information can be altered by the craft at any time. The recent-change data are managed by the recent-change-system process that is created immediately after the DSI/AP virtual circuit is established during software initialization. The recent-change process sends a download message to the AP asking for its tables that are stored on AP disk. If the AP recent change has reason to suspect that the recent-change information is not correct, it will ask DSI recent change to verify that the tables at a DSI are up to date. If these tables are out of date, the AP recent change will begin a download of all current information.

An update interface exists between the DSI recent change and its counterpart in the AP. When the craft enters a change at the AP, the updated recent-change tables are automatically sent to the DSI. The same procedures are followed as those for the original recent-change table insertion. Equipment status updates may require that an equipment remove or restore to service request be sent to the DSI equipment maintenance manager software.

4.3.5 DSI/AP clock synchronization

Accurate billing requires that the DSI be synchronized to the AP time-of-day clock. Time changes may be necessary because of DSI clock drift or changes to or from daylight savings time. A DSI time-change-system process manages corrections to the DSI time-of-day clock. Messages that exchange the time of day between the DSI and AP are used to update the DSI clock. The time-change process must note these changes in active billing records when the time of day is updated. Special attention must be given to changes that cross midnight boundaries. Timers that were set to expire based on the time of day must also be given special attention. A DSIOS primitive is provided to adjust these timers. The DSI clock can also be adjusted by craft command from the AP.

4.3.6 Emergency program update

Facilities are built into the DSI that allow the craft to examine and modify DSI data and text during in-service operation. Formatted octal or hexadecimal memory dumps of DSI memory or a snapshot of system process status information can be requested. A memory-patch facility allows emergency modifications to the operational software. This allows field updates without having to remove DSIs from service and without having to issue new software. Software patches are stored on AP disk and downloaded after the operational software is downloaded. Patches may be installed by the craft any time after the DSI is brought into service.

A software patch is installed by downloading the replacement software into a block of memory reserved for software patches. An assembly language jump instruction is inserted in the program text that is in error. This will transfer execution to the replacement software. The last executable statement in the replacement software will jump back to the executing program.

4.4 Data call processing

The data communication functions of the DSI are provided by the data-call-processing software. Data call processing controls the interfaces to subscriber access lines, the packet-switch access line, and provides text messages to subscribers during call setup. Each subscriber call is handled by a separate process called a call process. The call process performs both the data transfer functions specified in X.25 and LAPB, as well as handling the call setup and termination procedures. Each call process executes the same collection of shared protocol access programs. Per-call state information is kept in a data block that is allocated by DSIOS upon the call process creation. Each call process allocates a virtual call record for storing billing information. A call process is created when a subscriber initiates a request for service over a subscriber access line. Receipt of an X.25 LAPB frame initiates a request for service. This process will exist until the call is torn down. At that time, the call process will return any system resources to DSIOS and terminate itself.

Routing tables are used for associating packets received on an access line with the call process that handles the packet. A subscriber-line routing table is used to route all X.25 LAPB frames received on subscriber access lines to call processes. One entry exists for each line. Likewise, a network-channel routing table is used to route X.25 level-3 packets received on the packet-switch access line to the appropriate call process. One entry exists for each X.25 level-3 logical channel supported on this interface. Figure 14 shows a simplified view of packet routing and call-process architecture.

4.4.1 Call-control software

Coordination of call setup and termination procedures is performed by a program called call control, which is executed by the call process. Its responsibilities include allocating resources for the call when the call process is created, prompting the subscriber for a destination address, originating a request to X.25 level 3 to set up the X.25 virtual circuit, transmitting call progress messages to the subscriber terminal, coordinating call termination, and deallocating any system resources before the call process terminates.

4.4.2 Subscriber-access-interface software

The subscriber-access-interface software manages the software interface to the subscriber access lines. The main processor interface to subscriber lines is through the DMA processor. Interactions with the DMA processor are made through a hardware-control FIFO, a hardware status FIFO, and memory that is shared by both processors. The DMA processor interrupts the main processor when an entry is placed into an empty status FIFO. A DMA-processor-interface package provides a software interface to the DMA processor. Access routines are provided for transmitting control and data to the DMA processor. These routines are robust enough to allow use by both call processing and DMA processor diagnostics. Per-line status information is routed to the call process identified in the subscriber-line routing table.

The state transitions of dial-up line modems are controlled by DSI software. Modem status information is placed in shared memory by the control-buffer-and-clock circuit pack. A system process called the dial-up control process periodically scans this information looking for modem status transitions, such as ringing and carrier detected. When the dial-up control software detects that a subscriber is calling into

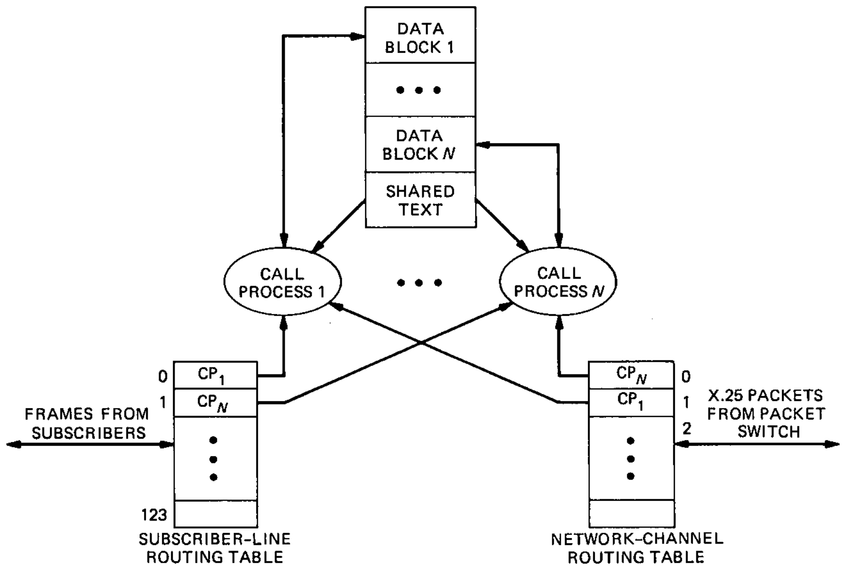

Fig. 14—Simplified diagram of call-process architecture.

the DSI, it takes the modem through the necessary handshake procedures for establishing an X.25 level-1 connection. Access routines are provided for external control of dial-up line states by other software systems.

A subscriber data-link manager called the customer-data-link status system process is responsible for serving events that may affect the status of subscriber access lines. For example, call waiting tones on a telephone line should not affect subscriber data service. The customer-data-link process must differentiate between the loss of the level-1 connection and the call waiting tones. If the link remains idle for more than a predetermined period of time, the customer-data-link status process assumes that the level-1 connection is lost and notifies the call process.

4.4.3 X.25 protocol controller interface software

The DSI supports an X.25 interface on the packet switch access line. A specially designed Large-Scale Integration (LSI) device called the XPC performs the entire X.25 level-2 protocol procedures. The X.25 level-3 protocol processing is performed by the call process. Because the X.25 level-3 processing is distributed across many call processes, a special system process, called the level-3 status process, globally monitors the status of the entire X.25 level-3 interface. Events such as packet-switch link failure, which affects all X.25 virtual circuits, are reported to this process. Handling of X.25 level-3 packets on the logical channel 0 is also performed by this process.

An XPC-interface software package, similar to the one used for the DMA processor, provides a software interface to the XPC device. Access routines are provided for transmitting level-3 packets and controlling the operation of the level-2 protocol. Priority queues are used to buffer packets sent by level 3 to the XPC device for transmission. The priority of a packet is specified in the packet buffer that contains the level-3 packet that is passed to the XPC-interface software. When the XPC device requests a packet for transmission, the highest-priority queue is serviced first.

The logical channel number of a received packet is used to index into the network-channel routing table for routing purposes. Any packet received on an inactive channel is routed to the level-3 status process.

4.5 Craft interface

The DSI craft-interface subsystem includes those programs that allow operations personnel to monitor and control the operations of the DSI. There are two methods that the craft can use to interact with the DSI. The Maintenance Terminal (MTTY) at the AP provides the craft with remote control of the DSI. Requests to execute hardware diagnostics, software audits, and requests to remove and restore DSI hardware can be sent to the DSI from the MTTY. The front panel on the power control and display circuit pack contains switches and LED indicators, which provide the craft with local control of the DSI and an indication of its current state.

4.5.1 Remote interface

Remote craft requests from the AP are routed to the DSI through the AP/DSI RICH communications package. There is no central DSI craft-input program. Craft requests are routed to the destination process by DSI-RICH software. Each destination process must then parse the input message to extract the information that is needed.

4.5.2 DSI power control and display panel administrator

Craft inputs can be made directly to the DSI through the Power Control and Display (PCD) panel circuit. Requests to remove or restore a line group or the entire DSI are made by activating switches on the panel. The PCD circuitry generates a hardware interrupt to the main processor when a switch is activated or a line group is powered on or off. When a PCD interrupt is detected, a PCD nonsystem process is created to handle the request in a background mode. This allows the DSI to continue processing calls.

When a line group is turned on, a PCD line-group diagnostic is automatically requested by the DSI maintenance software. Should the diagnostic fail, an alarm light is enabled on the panel and the line group is not brought into service. Otherwise, if the toggle switch associated with the line group is in the "restore" position, a request to restore the line group to service is made. Conversely, when a line group is turned off, a line group "remove request" is made by the PCD panel administrator.

The PCD software enables the "Request In Progress" (RQIP) light on the panel associated with the equipment. When the request has been completely processed the RQIP light is disabled, and the panel switch is examined to determine whether another request is pending.

The DSI has two 7-segment displays that are used to inform the craft of DSI status. Software-access routines are provided for controlling these displays. During system start-up, the PCD display is used to track the progress of DSI software initialization. After start-up is completed and the DSI is ready for normal operation, the display indicates the occurrence of certain errors.

4.6 Equipment maintenance

Equipment maintenance is the name given to those programs designed to control and monitor the operation and configuration of the DSI hardware. Equipment maintenance minimizes the impact of failures by providing the operating personnel with specialized hardware, software, and human interfaces to allow rapid detection, isolation, and repair of troubles. The specific tasks involved with equipment maintenance are restoring, removing, and diagnosing the DSI hardware, routinely exercising the system to ensure proper operation, and analyzing the system for problems.

4.6.1 Maintenance request administration

The central controller of equipment maintenance is a system process called the Maintenance Request Administrator (MRA). MRA coordinates equipment removal and restoral requests and hardware diagnostic requests. It also responds to inquiries on the state of equipment in the DSI. These requests can be made remotely from the MTTY, from the PCD panel, or from other DSI processes.

For nonstatus requests, MRA spawns nonsystem processes to handle each request. These processes are responsible for validating the request message parameters, determining whether the request can be honored given the current state of the machine, and returning a result message to the requesting process.

Requests to remove or restore equipment can be conditional or unconditional. Conditional requests for bringing equipment into service require running a diagnostic on the equipment. If the diagnostic fails, the request will be denied. Conditional requests to remove equipment from service will cause the MRA software to remove idle equipment and to camp-on busy lines. A camp-on procedure requires that the line be placed in a maintenance busy state. The MRA software will wait for the call to terminate before removing the line from service.

Unconditional requests for restoring equipment to service are handled immediately without running diagnostics. Unconditional requests to remove equipment from service cause any active calls on the affected lines to be terminated. Call termination messages are broadcast to the affected call processes to allow graceful call termination. After a brief delay to allow the broadcast messages to be transmitted, the lines will be removed from service.

When a diagnostic request is issued, the MRA determines which set of diagnostic phases are to be run. It then sends a diagnostic request message to the diagnostic software, which defines what phases to execute and how to execute them. Only one diagnostic request can be serviced at a time. If an additional diagnostic request is received, it will be queued until the currently executing diagnostic has completed.

4.6.2 Diagnostics

DSI diagnostics are those programs that test the DSI hardware and report detected faults. The goal in executing these diagnostics is to isolate errors to a single circuit pack as quickly as possible so that the faulty circuit pack can be replaced and the unit restored to full service.

To locate faults quickly, the diagnostics are organized so that they exercise the DSI hardware in layers. During the first stage, the first layer (the main processor and main memory) is diagnosed. Next, the other circuit packs in the processor complex are exercised, and finally, the group-distributor circuits and line cards are tested. By running diagnostics in this order, the core of the DSI is verified before any peripheral circuit packs are tested. Therefore, any errors uncovered are probably located in the circuit under test and are not the result of faults in a more central circuit pack.

Because the DSI is a simplex system, the diagnostics are divided into two categories: those that can execute without affecting the normal operation of the DSI and those that must be run when the DSI is out of service. Because the DSI downloads its operational software (including most of the diagnostics), it must have the capability to verify the minimal system needed to complete the download process before the download occurs. Therefore, some diagnostics are present in the bootstrap ROM for execution during a DSI boot sequence.

The diagnostic programs in the ROM- and RAM-based software systems consist of a two-part structure. A diagnostic controller schedules individual tests, reports errors, and performs the supervisory tasks needed to execute the diagnostics. The second part of the diagnostic structure consists of the diagnostic functions, or phases. Each phase contains multiple tests and is designed to exercise a portion of a particular circuit or subsystem. The first test of each phase, called the prologue, sets the environment for the hardware to be exercised. The last test of each phase, called the epilogue, restores the environment for normal operation when the diagnostic phase is complete. The phases contained in the operational software are organized so that a subset may be performed with the DSI in operation without affecting normal call-processing activities.

Diagnostic requests are received by the MRA, which subdivides the request into logical parts that exercise sections of the hardware at given intervals. After determining the logical breakdown, MRA creates the diagnostic-control process, which controls the execution of the diagnostic phases. Diagnostic failure messages are sent back to the AP MTTY. A completion message is sent back to the MRA software.

There are various modes of diagnostic operation that can be enabled in the diagnostic request at the AP MTTY. Running a diagnostic in "raw mode" causes each diagnostic failure to be reported back to the MTTY. Normally only the first failure is reported. The repeat mode allows the diagnostic to be executed more than once by a single craft command. The trouble-location-procedure mode causes diagnostic failures to be routed to software in the AP, which will output additional information about the probable cause of the diagnostic failure on the MTTY.

4.6.3 Routine exercises

The goal of the routine-exercises subsystem is to thoroughly verify the integrity of all hardware units of the DSI on a periodic basis without disrupting service to the subscriber. It detects hardware failures before they affect system performance. Routine testing is done during periods of low traffic. The routine-exercises subsystem includes line tests and the nondisruptive diagnostics of the common equipment. During routine exercises only lines without active calls are diagnosed.

Since the low traffic period of each DSI varies (observed by the operating personnel), the run time of the routine exercises can be changed depending on the low traffic period of the office. The routine-exercises trigger time is altered via recent change by the craft. Routine exercises can be disabled by recent-change input.

Routine exercises will first run in-service main-memory and main-processor diagnostics. Failures are reported to the craft. Next, routine exercises diagnoses each idle line. If a line diagnostic fails, a request is sent to MRA to remove the line from service. Craft will be notified that the line was removed. The number of lines that can be removed during a routine-exercises cycle is limited by a recent-change parameter.

4.7 System integrity

Service to the subscriber may be affected by hardware failures, software errors, data irregularities, and system overload. The system integrity software is designed into the DSI to minimize the impact of these failures on system performance.

4.7.1 Sanity detection

Basic system sanity is monitored by a hardware sanity timer. The timer is initialized by the start-up software and generates an unmaskable interrupt when it expires. Should the timer expire, software initialization is initiated. During normal operation, the system process called the sanity-timer reset process is responsible for periodically resetting the sanity timer. Should the system stop cycling for any reason, the sanity timer will automatically cause the DSI to recover without manual intervention.

4.7.2 Exception handler

Error handling is centralized in the DSI by providing a software interface through which all software and hardware errors are reported. This software subsystem, called the exception handler, is used to report and, when necessary, correct errors. The implication of each exception error may have different meanings for different processes. For example, certain processes may be able to recover from a temporary inability to allocate a system buffer, while others may not. To accommodate this flexibility, the exception-handler interface allows each process to pass a parameter that suggests the recovery action that should be taken. The final decision about error recovery is built into the exception handler for errors that have global system impact.

Each exception error has a unique error code. There are three categories of errors. The least severe are report errors. Software errors that do not affect system performance fall into this category. Report errors are reported to the AP and, in certain cases, a software audit is scheduled to run. Threshold errors can affect system performance if they happen too frequently. Each threshold error has a counter associated with it, which is incremented when the error occurs. Another system process, the error-analysis subsystem, periodically looks at these counters to determine whether further recovery action is required. The third category, called recovery errors, are serious errors that affect system performance and must be handled immediately. The DSI recovery subsystem is notified immediately to take corrective action.

4.7.3 Recovery

Recovery software is resident within the DSI to handle critical hardware and software faults that affect the entire system. Recovery takes immediate action to contain the failure before it becomes widespread. Since the DSI is a simplex system, hard faults cannot be circumvented and the affected equipment may have to be removed from service. Recovery actions taken include DSI boot, software initialization, packet-switch link reinitialization, line-group removal, or DSI removal. Errors are reported through the exception-handler software.

For each reset action, a report and alarm is issued to alert the craft about the failure. Should a recovery action require a DSI boot, the recovery software will store system status information in protected memory. This system snapshot, referred to as the postmortem, is transmitted to the AP for off-line analysis after the DSI boot has completed.

Neither the DSI recovery nor the DSI initialization invokes diagnostics directly. To find out more about the failing device, the AP craft must initiate a diagnostic session.

4.7.4 Error analysis

The main function of error analysis is to detect DSI hardware errors or abusive subscriber terminal conditions and isolate them before they can degrade DSI performance. Error counters maintained by error analysis are incremented by the operational software or through calls to the exception handler. The error counters are treated as "leaky bucket" integrators. Error analysis decrements them periodically and also checks to see if they have exceeded threshold. The decrement frequency and threshold of each counter is a function of the type of error condition which that counter is tracking.

Two types of error counters are maintained by error analysis. General errors are the result of faults from common equipment. When the threshold for any of the errors is exceeded, the recovery software is notified immediately to remedy the situation. Line errors can be isolated to a single subscriber access line or a line-card circuit. Line errors are pegged directly by the call-processing software. Depending on the type of line equipment and the error condition, a request to terminate a call can be sent to a call process if the line is active, or a request can be sent to the MRA to remove the line from service.

4.7.5 Audits

System audits protect against the loss of system integrity due to corrupted transient data. Transient data include routing tables and system resources such as packet buffers and message buffers. Audits are performed on a routine basis and upon demand. Data irregularities are corrected at the point of discovery.

The audit-controller process administers audit requests and schedules audits to run. Demand audits, requested by the DSI software through the exception-handler software, receive immediate attention by the audit controller. These requests are made when certain errors indicate a possible loss of resources or when the number of available key resources drops below a system threshold. Manual audits, requested by the craft, can be scheduled at any time but are queued behind demand audits. The lowest-priority audits are routine audits. They are scheduled frequently enough to support the main-line execution, but not so frequently as to affect system capacity.

For every irregularity detected, a report is generated that provides enough information about the irregularity for off-line analysis. Because a single data irregularity may have multiple side effects that could flood the AP with many reports, the number of audit reports is limited for each invocation of the audit controller.

4.7.6 Overload control

The objectives of the overload-control subsystem are to smooth variations in traffic loads, to provide normal service to existing subscribers, and to prevent an overall system outage due to overload. The overload controls are aimed primarily at controlling packet-buffer congestion. The purpose of packet-buffer congestion procedures is to avoid the negative consequences of running out of packet buffers. Running out of packet buffers could result in calls that are deadlocked or prematurely dropped. Overload monitors are built into the main-line code. When critical resources drop below defined thresholds, overload actions are taken. Several levels of control are allowed by having two overload states. Hysteresis is used to smooth the transitions between overload states. Overload controls include flow controlling access lines and blocking new call originations.

4.8 DMA-processor firmware

In addition to the software that runs on the DSI main processor, the DMA processor is controlled by its own firmware, which resides in a separate ROM on the DMA-processor circuit pack. This firmware is designed to operate strictly in response to external events, and it includes both operational and maintenance programs. The DMA-processor operational software is an extremely compact and efficient set of assembly language routines. These routines are invoked by a main routine, which cyclically scans the FIFO memory interfaces to the main processor and the multiplexed protocol formatter for entries. The scanning procedure has been designed to optimize the DMA-processor's response time to critical events. When an entry is detected in one of the FIFOs, the DMA processor reads the entry, takes appropriate action, and then returns to the main scanning routine. In addition to the operational software, the DMA processor contains certain limited diagnostics and maintenance software. Many of the maintenance routines are divided into short sections, each of which executes very quickly. This permits these routines to run while the DMA processor is processing customer information without affecting service.

4.9 Data flow through the DSI

In order to help clarify the functions of many previously described DSI software subsystems, and in particular the data-call-processing subsystem, this section traces the progress of a data call through the DSI. The treatment is developed for a dial-up data call, although the sequence of events for a dedicated access data call is similar, but does not require a level-1 connection to be established.

4.9.1 Establishment of a level-1 connection on dial-up lines

When a subscriber dials the LADT access number, the public telephone network routes the call to a line in a line hunt group to which a DSI is connected. The terminating voice switch applies ringing to that line. The modem line circuit detects the presence of ringing and sets an internal-status bit. When the dial-up control-unit software scans these bits and detects ringing, it commands the modem line circuit to go off-hook and answer the incoming call.

After two seconds, the dial-up control-unit software commands the modem line circuit to send an answer tone. When the subscriber modem receives an answer tone, it should respond by sending carrier. If the subscriber terminal does not send carrier within 18 seconds, the modem line circuit will go back on-hook. When the dial-up-control-unit software detects carrier, it commands the modem line circuit to send an answering carrier.

After 1 second, the multiplexed protocol formatter will start transmitting flags on that line. A packet buffer is allocated for receiving the first LAPB frame from the subscriber terminal. If the LAPB protocol is not established within a specified period of time after flags are sent, the dial-up control-unit commands the line to go back on-hook.

4.9.2 Call set-up/termination flow

For a dial-up subscriber line, the DSI must go through a modem handshake sequence to establish the level-1 (circuit-switched) connection. For in-service dedicated lines, the DSI is always sending flags to the terminal indicating a willingness to establish a level-2 connection. After a level-1 connection is established, the subscriber may request service by establishing the X.25 level-2 LAPB protocol. When a frame is received on a subscriber line whose entry in the subscriber-line routine table is free, the DMA processor interface software creates a call process. The routing table is updated to point to the newly created call process, and a message identifying the subscriber line number and the received frame are forwarded to the call process.

The first frame expected by the DSI LAPB protocol is a Set Asynchronous Balance Mode (SABM) command for establishing the level-2 protocol. Non-SABM frames are discarded and the call process is terminated. Otherwise, an acknowledgment is transmitted to the terminal and the call-control program is invoked. Call control transmits an ASCII prompt to the terminal requesting a destination address. These signaling messages between call control and the terminal are transmitted in LAPB information frames. If a destination address is not received within a specified period of time, call control will terminate the call process.

When a valid address is received from the user, call control selects a free logical-channel number entry from the network routing table. The routing table is updated to associate the call process with the selected logical channel. Call control creates an X.25 level-3 *call-request* packet and invokes the X.25 level-3 program to set up a call through the packet switch. A *call-attempt signaling message* is transmitted to the terminal informing the user that the call is being placed. Call control waits for an acknowledgment from level 3 that a virtual call has been set up and periodically sends the *call-attempt signaling message* to the terminal. When the call-connect packet is received, it is passed to call control for a facility field verification. If the parameters are valid, a *call-connected signaling message* is sent to the user's terminal. After the call is set up, all data are passed transparently on the virtual circuit.

A call can be terminated by the subscriber terminal, remote host DTE, or by the LADT network due to internal errors. A subscriber terminal can terminate a call by transmitting a LAPB Disconnect Command (DISC) to the DSI or by terminating the level-1 protocol (hanging up). The remote host DTE terminates the call by clearing the virtual circuit. The DSI can initiate call termination by either of the above methods.

When the DSI receives a DISC command from the subscriber terminal, an acknowledgment is sent to the terminal and the subscriber-line routing table is restored to the free state. Call control commands level 3 to clear the virtual call. The call process will wait for an acknowledgment (or time-out) from the packet switch. When the acknowledgment is received, the network-channel routing-table entry is restored to the free state. The billing record is sent to the billing process and the call process terminates.

4.9.3 Call flow for packets received from terminals

The largest percentage of the DSI main-processor real time is used to switch ordinary information packets after the data call has been established. The DSI data-call processing has been designed to be real-time efficient in effecting such packet transfers. The paragraphs that follow describe the logical steps of the hardware and software interaction for a packet received from a subscriber terminal.

When the multiplexed protocol formatter detects the start of a packet, it builds up the first byte of the packet by accumulating the data bits. After it has assembled an entire byte, it transfers the data byte to the DMA Processor. The DMA processor and the main processor communicate by using bidirectional hardware FIFOs for time-critical events, and by using a shared memory accessible by both the DMA processor and main processor for routine events. When the DMA processor receives the first byte of a packet, it looks into its shared memory for the address of the packet buffer that has been previously allocated for that line. If a valid packet-buffer address is present, the DMA processor transfers the byte into main memory at the address specified by the packet buffer. The DMA processor also puts a message into its FIFO for the main processor to indicate that it has started to receive a packet from the subscriber line. As the multiplexed protocol formatter assembles additional bytes, it passes them on to the DMA processor, which in turn transfers the bytes into the packet buffer in main memory.

While the packet is being transferred into main memory, the DMA-processor interface software reads the FIFO from the DMA processor and discovers that the subscriber line has started to receive a packet. The DMA processor will need a new packet buffer after the packet reception is complete, so a new packet buffer is allocated and its address is written into the DMA processor's shared memory. This preallocation of packet buffers is necessary so that another packet can be received immediately after the end of the current packet is detected.

When the multiplexed protocol formatter detects the end of the packet, it confirms that the reception was accurate by checking the CRC at the end of the packet and then informs the DMA processor that the packet reception has been completed. The DMA processor then writes an entry into the FIFO to the main processor indicating that the subscriber line has just received a packet with a good CRC.

After the DMA-processor interface software reads this entry from the FIFO, it routes the packet to the call process associated with the subscriber line. All inputs to a call process are funneled through a single entry point that is waiting for messages. When a LAPB frame is received, the LAPB protocol is invoked to process the frame. The LAPB program will transmit responses to control frames while data frames are passed on to the X.25 level-3 program for transmission to the packet switch.

The X.25 level-3 program queues data received from the subscriber terminal if the virtual circuit to the remote DTE has been flow-controlled or the flow-control window is full. Otherwise, it inserts an X.25 level-3 data-packet header and calls a packet-switch access routine to transmit the packet. The XPC-interface software queues the packet for transmission based on the priority of the packet specified in the packet buffer by level 3. The XPC device notifies the main processor when X.25 level-2 acknowledgments for a packet are received form the packet switch. This allows the XPC-interface software to deallocate the packet buffer.

4.9.4 Call flow for packets received from packet switch

The packet flow through the DSI from the packet switch is similar to that for packets received from a subscriber terminal and is not covered in as much detail. The hardware, firmware, and software interaction for transmitting a packet is also similar to that required for receiving a packet and is not elaborated in detail. The XPC device interrupts the main processor after an incoming packet has been placed into main memory by DMA. The XPC-interface software driver extracts the X.25 level-3 logical-channel number from the level-3 header for routing purposes. If the routing-table entry is free or invalid, the packet is routed to the level-3 status process. Otherwise, the packet is routed to the appropriate call process. The call process will invoke the level-3 program when it receives a packet from the packet switch. When level 3 determines that a valid data packet has been received, it invokes the LAPB protocol program to transmit the data out to the subscriber terminal.

LAPB will queue the data if it is flow-controlled or the flow-control window is closed. Otherwise, it inserts a LAPB header and invokes a DMA-processor-interface software transmit routine which queues the frame for transmission. Each subscriber access line has an output transmit queue. If the transmit queue is empty, the packet buffer address is placed in shared memory and a command is sent to the DMA processor to start transmission. If other frames are queued, the frame is placed at the end of the transmit queue. The DMA processor will interrupt the main processor when it has completed a transmit request. The DMA-processor-interface software will then initiate another transmission on that line if a frame is queued.

V. ADMINISTRATIVE-PROCESSOR SOFTWARE ARCHITECTURE

The AP is implemented under the DMERT (Duplex Multi-Environment Real Time)[13,14] operating system that runs on the AT&T 3B20D computer.[15] The 3B20D computer was chosen for its high reliability. All AP software is written in C programming language, most of it residing at the user level of the operating system, with only a few real-time critical functions running at the kernel-process level. Figure 15 shows the internal architecture of the AP software.

5.1 X.25 handler

The AP communicates with the rest of the LADT network over one 9.6-kb/s link running levels 2 and 3 of the X.25 protocol. The AP-X.25 implementation is based on the DMERT-supplied X.25 package. The DMERT package was modified to support dynamic virtual calls and to conform to the version of X.25 supported by the packet switch.[16] The LADT implementation of X.25 retains the basic file interface for the *UNIX** operating system provided by DMERT. However, a new interface was added to notify AP processes that use X.25 about changes in the status of X.25 calls. Each of these AP processes attaches to its own DMERT message port, to which the X.25 handler sends a DMERT message whenever a call to that process is set up or cleared.

Since the DSIs all use identical copies of the DSI software, a DSI has no way of knowing which DSI it is. However, the packet switch can identify a DSI by the physical link on which the DSI call originates. If the DSI does not fill in the originating address in the call request packet, the packet switch will fill in an originating address unique to that DSI. The AP can then use this originating address to identify the DSI. The table of legal DSI addresses is stored in the DMERT-supplied equipment configuration database, and the AP will refuse any call that is not from one of these addresses.

5.2 Generic download

Since the DSIs are RAM-based, their operational generics must be downloaded from the AP. There is a ROM-based start-up program in the DSI that includes some hardware diagnostics, a partial X.25 level-3 implementation, and the DSI portion of the generic download

* Trademark of AT&T Bell Laboratories.

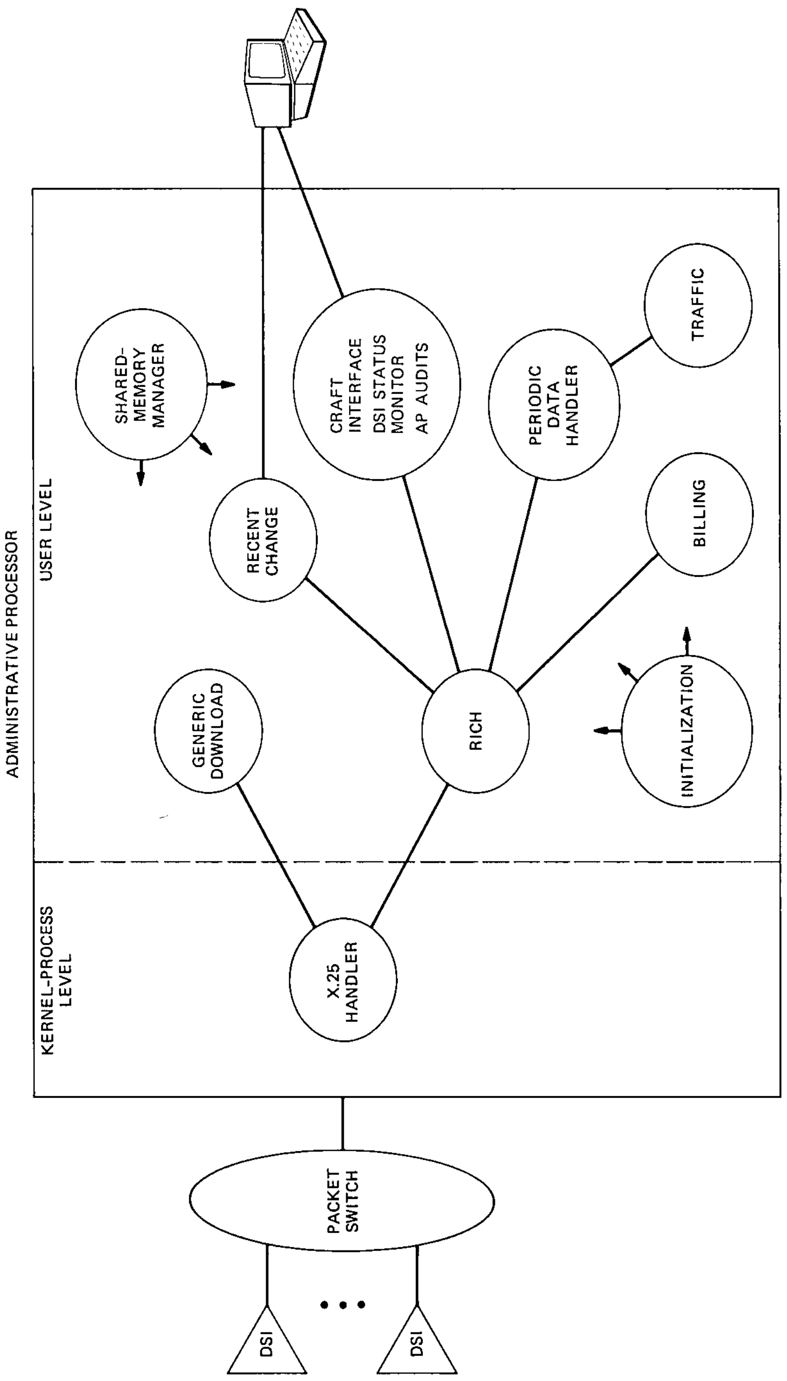

Fig. 15—Internal architecture of the administrative-processor software.

program. At start-up, the DSI first runs its internal diagnostics and then sets up a call to the generic download process in the AP.

When the AP receives a request from a DSI for a download, it first checks a history file to see when the DSI was last downloaded. If the DSI has requested an excessive number of downloads within a specific period of time, the AP will alert the craft and refuse to download the DSI. If the download request is accepted, the AP generic download function checks the LADT database to determine which DSI generic should be sent. The generics are divided into 1792-byte blocks, with a checksum over each block. When the DSI receives each block, it computes the checksum, and then sends an acknowledgment back to the AP for that block. The AP end of generic download will only transmit blocks to the DSI as long as there are fewer than five unacknowledged blocks outstanding. This windowing mechanism is used so that the AP does not have to always idly wait for the acknowledgments, but will also not needlessly send several bad blocks after an error has occurred. If the DSI does report that there was an error in receiving a block, the AP will resume sending the generic starting at that block. If the download is interrupted for external reasons such as a failure of the DSI-to-packet switch link, the DSI generic download program keeps track of the last good block that it received and notifies the AP to start at that block number when it re-establishes the call.

5.3 Shared-memory manager

When large amounts of data must be moved between two processes, one of the most efficient methods is through the use of shared memory segments. The shared-memory manager is a process that centralizes the handling of shared-memory segments in the AP. When the AP is initialized, the shared-memory manager reserves all the shared memory segments using DMERT's *UNIX* system calls. Each shared memory segment has a process associated with it that is considered to be the master of the segment. For example, the RICH process controls the shared memory that is used to pass RICH transactions between it and application processes.

When the AP reinitializes, it is not known which processes will need to be restarted. Because of this, it is conceivable for user processes to request to be attached to shared memory before the memory has been initialized by the master process. To circumvent this possibility, the shared-memory manager waits until the master process for each segment notifies it that the segment has been initialized before it allows the other processes to attach to the shared-memory segment.

Besides controlling the initialization sequence, the shared-memory manager also allows a greater degree of fault tolerance in the AP architecture. If each master process reserved its own segment using DMERT's *UNIX* system calls and the master process terminated, it would lose track of the memory segment and all the data stored in it would be lost. With the shared-memory manager, if a process terminates, it can ask the shared-memory manager for the correct memory segment when it reinitializes. Since the shared-memory manager is a simple process, it is considered less likely to terminate than some of the more complicated processes.

5.4 Remote internal communications handler

The only way the AP can communicate with the DSIs is over X.25 virtual circuits. However, there are many processes in each DSI that the AP needs to address, such as maintenance, recent change, billing, and traffic. If one virtual circuit was assigned between the AP and each DSI for each individual process that needed to be addressed, the AP would have to terminate several hundred virtual circuits.

To avoid having to terminate many virtual circuits in the AP, a new internal protocol, RICH, was developed, which allows most of the AP-to-DSI communication to share one virtual circuit between each DSI and the AP.

A RICH transaction consists of a 12-byte header and up to 244 bytes of user data. RICH addresses are 4-byte integers that allow processes to be addressed by a combination of their function (such as billing) and the type and instance of their host processor (such as DSI number 3). Within a given process address, up to 100 RICH transaction types may be locally defined. For instance, the maintenance processes could use different transaction types to distinguish between different types of diagnostic messages that are returned from the DSIs. A destination process can ask to be given either the oldest message of a given type or just the oldest message of any type.

The RICH in the AP is implemented using a dedicated memory segment that is shared among all the processes that use the RICH. The RICH provides function calls to send and receive packets which are implemented as a *UNIX* software library that is built into every process that uses the RICH.

One complication in the RICH is that the byte and word orders of the AP and the DSI processors are different, making it impossible to pass data structures directly from one processor to the other. To pass data structures other than arrays of characters between the two processors, every data structure must have the bytes "swabbed" to the correct order for the other processor.

5.5 Periodic data handler

Every 5 minutes, the DSIs each send traffic counts to the AP in a RICH transaction. Since these counts are needed by several AP processes, another shared-memory segment was created to allow all the processes to access the same copy of the data. The periodic data handler is the process that reads the traffic transactions from the RICH and puts them into the shared-memory segment. The periodic data handler uses data in the recent-change table and from the DSI status monitor to determine from which DSIs it should receive transactions. When transactions have been received from all the DSIs, it sends a signal to all the AP processes that use the data. If the transactions from all the DSIs have not been received within a short period of time after they are expected, the periodic data handler informs the DSI status monitor that there may be problems with a DSI and signals all the user processes to read the data.

5.6 Billing process

The DSIs send billing transactions to the AP using RICH transactions. While several transactions may be sent for very long calls, the AP does not do any call record assembly. The DSIs also send hourly tracer records that contain counts of the number of billing records of each type that the DSI has sent. The AP billing program checks the number of records that it has received against the counts given in the DSI tracer records. Any errors in the counts are then reported in tracer records generated by the AP for the billing processing center. For most telephone operating companies, the bill generation will be done by revenue accounting offices.

The processing of billing records in the AP is done in two stages. As records arrive from the DSIs, they are processed to check their validity and to calculate holding times. Then that information along with other billing data such as packets sent and received based on rate periods is written to a daily billing file stored on disk. Both dial-up and dedicated records are read by this process, but only dedicated records are processed. Both types of records are also written to a log file that allows off-line analysis of the per-call data. The last five daily billing files are stored on disk, allowing several days worth of billing data to be written to tape at the same time. If errors are encountered during the reading of the tape at the revenue accounting office, a new copy of the tape can be generated from the disk data.

5.7 Traffic data processing

Every 5 minutes, each DSI sends a block of traffic data to the AP in a RICH transaction, which is then read into the periodic data handler. The traffic report generator then reads the traffic data out of the periodic data handler to generate reports. The traffic process prepares reports at 5-minute, 30-minute, and 24-hour intervals with identical content, though it is possible to tailor the printed output of the different reports through the use of recent-change forms. For instance, the output printed in the 30-minute reports can be limited to the subset of the reports that is needed on a timely basis. The entire report could then be printed only at midnight. The 5-minute reports can be turned on conditionally so they will be printed only if the value in one of the report fields exceeds a set threshold. Thresholds can be set on any field in the report through recent change. Crossing a threshold also causes an error message to be printed. Threshold exceptions and any other DSI errors are summarized in 5-minute, 30-minute, and 24-hour error reports that can also be tailored through the recent-change system.

5.8 Recent change

The AP recent-change system stores all configuration data for the AP and the DSIs. While the AP stores the master copy of all configuration data, the recent-change data for each DSI is sent to the DSI when the DSI is initialized or when the copy in the AP is changed. The craft interface to the AP recent-change system is through the DMERT-provided Online Data INtegrity (ODIN) system, a forms-based data entry system that provides several levels of data validation. Many new ODIN forms were created for LADT, including forms for traffic report formats, DSI equipment forms, DSI subunit equipment forms, and customer data forms.

The AP recent-change database is implemented using a shared memory segment and a *UNIX* software library of recent-change function calls. For the simple types of data retrieval needed in LADT, a shared-memory implementation is faster than a database built using the more powerful DMERT-supplied database tools.

5.9 DSI status monitor

While the recent-change system stores data on the DSIs that are equipped, a different mechanism is needed to store the maintenance state of the DSIs. The DSI status monitor provides the current maintenance status of the DSIs to both internal AP processes and to the craft. The DSI status monitor classifies each DSI in one of the following states:

1. Active
2. In the process of being downloaded
3. Manually out of service
4. Link down
5. Out of service.

The DSI status monitor gathers data on the state of the DSIs from many sources. First it determines which DSIs are equipped from the recent-change system. When a DSI starts a generic download, the

X.25 handler notifies the DSI status monitor that the DSI is in the download state. When the download is finished and the DSI operational, the DSI sends a message to the DSI status monitor stating that it is active. The DSI status monitor checks the periodic data handler every 5 minutes to see which of the DSIs have sent data for that 5-minute period. If the periodic data handler informs the DSI status monitor that data were not received from a DSI that should be active, the DSI status monitor sends a RICH transaction to the DSI querying the DSI status. If the DSI status monitor does not get the proper response from the DSI, it marks the DSI out of service and triggers an alarm. The X.25 handler also alerts the DSI status monitor any time a call from a DSI to the AP is torn down.

5.10 DSI craft interface

While some local craft capabilities are provided by the DSI power control and display panel, most of the craft interface for the DSIs is provided through the AP. All LADT craft commands conform to the Program Documentation Standards (PDS) formats used for many types of AT&T switching systems. Some commands execute locally on the AP, while other commands are entered on the AP and then sent to a particular DSI for execution. Where the commands actually execute is transparent to the craft who enter the commands.

5.11 AP maintenance

Besides the LADT maintenance functions for DSIs, the AP has some internal maintenance functions to augment the maintenance functions provided by DMERT. The AP uses the User Level Automatic Restart Process (ULARP) provided by DMERT to handle internal initializations. This process has a list of AP processes that should be started at initialization and restarted if any of them terminate. To prevent thrashing, the application integrity monitor checks how often a process is restarted. If any process is restarted more than twice in 6 minutes, the application integrity monitor will cause an AP software initialization.

The AP has one level of application initialization defined below the DMERT levels of initialization. If fatal errors are encounted in running the AP application code, this application level of initialization restarts all the LADT specific processes without restarting DMERT. If two iterations of user-level initialization occur within a 5-minute period, then the initialization level is escalated to DMERT initializations.

5.12 AP audits

Since many AP processes use shared memory segments, it is important to prevent a bad process from corrupting the data stored in the segments. The AP has internal audits for the shared memory buffers that run periodically. These audits attempt to correct any errors that are found, or cause a software initialization if the errors cannot be corrected.

VI. SUMMARY

The LADT system has been designed to provide data-transport services within a Local Access and Transport Area (LATA). AT&T Technologies implementation of LADT divides the necessary functions among already existing and newly designed components in order to provide economical access for various types of customers. The network has been designed to be both cost-effective and reliable.

REFERENCES

1. H. J. Kafka et al., "Local Area Data Transport—A Packet-Switched Network For Exchange Area Services," IEEE Int. Conf. on Commun. ICC '83, June 1983.
2. G. J. Handler, "Networking—Bit by Bit," Conf. Record of Videotex '82, pp. 453–63.
3. C.C.I.T.T., Data Communication Networks Services and Facilities, Terminal Equipent and Interfaces, Vol. VII–Fascicle VIII.2, Geneva, 1981.
4. M. N. Ransom, "Local Area Data Transport System Overview," AT&T Bell. Lab. Tech. J., this issue.
5. W. L. Harrod and A. G. Lubowe, "The *BELLPAC*™ Modular Electronic Packaging System," B.S.T.J., *58*, No. 10 (December 1979), pp. 2271–88.
6. A. B. Glaser et al., "The XPC—A VLSI Link-Level Controller for X.25 LAPB," IEEE Int. Conf. on Circuits and Computers ICCC '82, September–October 1982.
7. N. E. Snow and N. Knapp, Jr., "Digital Data System: Sytem Overview," B.S.T.J., *54*, No. 1 (May 1975), pp. 811–32.
8. E. C. Bender, J. G. Kneuer, and W. J. Lawless, "Digital Data System: Local Distribution System," B.S.T.J., *54*, No. 1 (May 1975), pp. 919–42.
9. P. Benowitz et al., "Digital Data System: Digital Multiplexers," B.S.T.J., *54*, No. 1 (May 1975), pp. 893–918.
10. "Data Set 212A Interface Specification," AT&T Technical Reference, PUB 41214, January 1978.
11. J. R. Boddie et al., "Digital Signal Processor: Architecture and Performance," B.S.T.J., *60*, No. 7, Part 2 (September 1981), pp. 1449–62.
12. "Local Area Data Transport Terminal Interface Specifications," AT&T Technical Reference, PUB 54200, June 1982.
13. J. R. Kane, R. E. Anderson, and P. S. McCabe, "Overview, Architecture, and Performance of DMERT," B.S.T.J., *62*, No. 1, Part 2 (January 1983), pp. 291–302.
14. M. E. Grzelakowski, J. H. Campbell, and M. R. Dubman, "DMERT Operating System," B.S.T.J., *62*, No. 1, Part 2 (January 1983), pp. 303–22.
15. W. N. Toy and L. E. Gallaher, "Overview and Architecture of the 3B20D Processor," B.S.T.J., *62*, No. 1, Part 2 (January 1983), pp. 181–90.
16. "X.25 Interface Specifications," AT&T Preliminary Technical Reference, PUB 54010, August 1981.

GLOSSARY

AP	administrative processor
CRC	cyclic redundancy check
DMA	direct memory access
DISC	LAPB disconnect command
DMERT	duplex multi-environment real time

DSI	data subscriber interface
DSIOS	DSI operating system
DSP	digital signal processor
DTE	data terminal equipment
FIFO	first-in first-out
FSK	frequency shift carrier
LADT	local area data transport
LAPB	link access procedure B
LATA	local access and transport area
MOS	metal oxide semiconductor
MRA	maintenance request administrator
MTTY	maintenance terminal
ODIN	on-line data integrity
PCD	power control and display
PDS	program documentation standards
RAM	random access memory
RICH	remote internal communication handler
ROM	read-only memory
RQIP	request in progress
SABM	set asychronous balance mode
SLC	subscriber loop carrier
XPC	X.25 protocol controller

AUTHORS

Henry J. Kafka, B.S. (Electrical Engineering), 1979, Northwestern University; M.S. (Electrical Engineering), 1980, University of Illinois, Urbana; AT&T Bell Laboratories, 1979—. At AT&T Bell Laboratories, Mr. Kafka's responsibilities have included hardware and software design, as well as applied research in new digital services and advanced packet-switching concepts. Mr. Kafka is presently Supervisor of the LADT System Evaluation Group and is responsible for software development and field support for AT&T's Local Area Data Transport products. Member, Eta Kappa Nu, Tau Beta Pi, IEEE.

W. Joseph Paule, B.S. (Computer Science), 1976, Iowa State University; M.S. (Computer Science), 1977, University of California, Berkeley; AT&T Bell Laboratories, 1976—. Upon joining Bell Laboratories, Mr. Paule worked on the development of the 1A Voice Storage System (1A VSS), participating in tool development, system integration, system test, field support, and feature design. He was promoted to Supervisor in 1980. From 1981 to 1983, Mr. Paule worked on the LADT project, where he supervised software development, system integration, and field support. He is currently in the Data Packet Switching Department, where he is Supervisor of the 1PSS System Test and Field Support Group. Member, Phi Beta Kappa, Phi Kappa Phi, ACM.

David J. Stelte, B.S., Electrical Engineering (summa cum laude), 1972, University of Notre Dame; M.S. (Electrical Engineering), 1973, University of Illinois; GTE Network Systems Digital Switching Development Laboratory, 1974–1978; AT&T Bell Laboratories, 1978—. At GTE, Mr. Stelte was involved in the design and implementation of Pulse Code Modulation systems as well as system architectural work on class 5 digital switching systems. At AT&T

Bell Laboratories, Mr. Stelte's responsibilities have included applied research in new digital services, and work on ISDN standards and their implementation in digital switching systems. He has also worked in the areas of data set design and mobile telecommunications. Mr. Stelte is presently Supervisor of the LADT System Design Group and is responsible for architectural planning and software development for AT&T data switching products. Member, Eta Kappa Nu, Tau Beta Pi, IEEE.