COMSAT Laboratories

FINAL REPORT

COMSAT PARTICIPATION IN DARPA PACKET SATELLITE INTERNETWORKING AND SPEECH APPLICATIONS PROGRAM

Sponsored by

DEFENSE ADVANCED RESEARCH
PROJECTS AGENCY
ARPA ORDER NO. A 0.3774

Monitored by

DEFENSE SUPPLY SERVICE
WASHINGTON (DSSW)
UNDER CONTRACT NO.
MDA 903-79-C-0308

July 1980

COMMUNICATIONS SATELLITE CORPORATION
COMSAT Laboratories, Clarksburg, Maryland 20734

FINAL REPOR

AT PARTICIPATION IN DARPA
SATELLITE INTERNETWORKING
ECH APPLICATIONS PROGRAM

Sponsored b

DEFENSE ADVANCED RESEARCH
PROJECTS AGE CY
ARPA ORDER NO. 0774

Monitored

EFENSE SUPPLY ERVICE
WASHINGTON (DSSW)
UNDER CONTRA T NO.
MDA 903-79-C-0308

AUG 19 0

DISCLAIMER NOTICE

THIS DOCUMENT IS THE BEST QUALITY AVAILABLE.

COPY FURNISHED CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

(1)

WP2353/KAC44A

COMMUNICATIONS SATELLITE CORPORATION
950 L'Enfant Plaza, SW
Washington, D.C. 20024

TSLAP-80-6 Final Report

Issued July 1980
Sponsored by
Defense Advanced Research Projects Agency
ARPA Order No. A:0.3744

Monitored by DSSW Under
Contract No. MDA 903-79-0308

Period of Performance
15 March 1979 to 15 June 1980

L. C. Palmer
Program Manager
(301)428-4575

ACKNOWLEDGMENTS

This final report was prepared by various staff members at COMSAT Laboratories. Most of the work under Task 1 has been performed by S. Rothschild and J. Thomas of the Communications Processing Laboratory (CPL). Mr. Rothschild provided the material in Subsections 2.2 and 2.3. J. McCoskey helped throughout the contract period in SATNET performance data analysis, C&M module checkout, and the development of the wideband timing simulation program. L. Mikisits also helped with the SATNET data summaries and prepared the compilations in Subsection 2.4.

Dave Mills and Hoi Chong were the major participants in Task 2, Internetworking Experiments, throughout the contract period. The success of the NTC demonstration was largely due to their diligent efforts. Dr. Mills prepared the material in Section 3 and Appendices A and B.

J. Kaiser has been involved in the ARPA program since fall 1979, first in helping with the logistics relative to the NTC demonstration, and recently in assuming the lead role in COMSAT's coordination/integration of the wideband system.

Ross Snider, who is on temporary assignment at COMSAT Laboratories, provided an independent assessment of the C&M capability and prepared Section 5 of this report.

The contributions of all of these individuals during the past 15 months are acknowledged.

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

Table of Contents (continued)

List of Illustrations

List of Illustrations (continued)

List of Tables

1. INTRODUCTION/SUMMARY

1.1 INTRODUCTION

This final report, submitted by the Communications Satellite Corporation (COMSAT) under Contract MDA 903-79-C-0308, describes COMSAT's technical activities and accomplishments from March 1980 to June 1980, under three tasks: Task 1, Transition of SATNET to Operational Status; Task 2, Internetworking Experiments; and Task 3, Wideband Domestic Network Coordination/ Integration. It also summarizes the activities and accomplishments under the three tasks during the total contract period (March 1979-June 1980).

During this reporting period, an internetting/SATNET meeting was held at MIT where COMSAT reported on progress under Tasks 1 and 2 of the contract. Progress was also reported on the installation of the third PSP terminal in Tanum, Sweden, during the first half of May 1980.

1.2 TASK SUMMARIES

1.2.1 TASK 1: TRANSITION OF SATNET TO OPERATIONAL STATUS

During the early part of the contract period, COMSAT's participation in the Task 1 activities was limited by other higher priorities. For example, the checkout, delivery, and installation of the three Packet Satellite Program (PSP) terminals, which were

completed October 31, 1979, under a previous contract, overlapped this period and thus delayed the start of some Task 1 activities. Preparations for the National Telecommunications Conference (NTC) demonstration in November 1979 also made it necessary to postpone upgrading the prototype PSP terminal that is used with the unattended earth terminal (UET) at Clarksburg. In addition, command and monitoring (C&M) module calibration was delayed until after the NTC demonstration.

The highlights of Task 1-related activities during this period are as follows:

June 1979	PSP terminal delivered and installed at ETAM
September 1979	PSP terminal delivered and installed at Goonhilly
September 1979	Decision to retain third PSP terminal (scheduled for delivery to Tanum) at COMSAT Laboratories for an indefinite period; this terminal would be used to support the NTC demonstration and afterward to measure and calibrate the C&M functions including the T&M parameters
December 1979	Plan developed for the PSP terminal maintenance, including the formation of a facility at COMSAT's Maintenance and Supply (M&S) Services Center, and the procurement and/or fabrication of spare parts; this plan was submitted as a proposal to DARPA.
March 1980	Filing submitted to the FCC for SATNET transition from experimental to operational status commencing at approximately mid 1980.
March 1980	C&M calibration completed on the Tanum PSP terminal

During the final contract reporting period (March-June 1980), the major effort in Task 1 was devoted to the delivery and installation of the third PSP terminal, which was shipped on April 18, 1980, and installed May 8 to 16. During installation, problems were discovered which required the return of a terminal oscillator to the original supplier. After about a week of erratic operation of the Tanum PSP terminal, the original SPADE system was restored in early June. However, additional complications arose which caused the erratic behavior to continue. A continuous effort involving frequent direct contact with the Tanum earth station has been necessary to the end of the reporting period to isolate and resolve these problems.*

After the Tanum installation was completed, COMSAT personnel visited Goonhilly to examine the C&M module. The most obvious problem, the inability of the SIMP to communicate with the C&M module, was traced to a broken wire which was repaired quickly. Afterward, the same symptoms were observed at Goonhilly as at the other stations, namely, a high packet miss rate by the SIMP on certain links when the T&M parameters were enabled. A test performed at Goonhilly isolated the problem to an incompatible format between the T&M words and the SIMP hardware. Alternative solutions to this problem are presently being reviewed (see Subsection 2.2).

Another major effort during the reporting period was the integration and checkout of the C&M module into the prototype PSP terminal being used with the UET at Clarksburg. This equipment was used during early experiments and was generally incompatible with the addition of the C&M module and its interconnections.

*As of the end of July 1980, the repaired oscillator has been returned to Tanum and reinstalled in the PSP terminal. This coincided with SIMP maintenance by BB&N and resulted in normal operation of the Tanum PSP in SATNET.

Thus, a complete rework of the prototype terminal was required. The checkout of this refurbishment was completed near the end of the contract period.

One major long-term subtask under Task 1 was the evaluation of C&M functions which are now part of the new PSP terminals. Section 5 describes the available C&M functions and states conclusions concerning the relative usefulness of the different capabilities. Recommendations are also made about the initial processing of the T&M parameters which are an important subset of the C&M capability. It should be recognized that this evaluation is perhaps premature, since very little operational experience has been collected. The potentially most useful part of the C&M module, the T&M parameters produced by the modem on each packet, has not been available because of the data format incompatability noted above.

1.2.2 TASK 2: INTERNETWORKING EXPERIMENTS

A major effort during the first half of the contract period was devoted to preparations for a demonstration of the SATNET technology which was to culminate at a session at the NTC held in Washington, D.C., on November 27-29. The demonstration included packet speech and facsimile transmission between the conference site and University College London (UCL). COMSAT coordinated the necessary hardware and software development between the major participants. The NTC demonstration followed many months of preparation and the resolution of a variety of technical and non-technical problems. The goal of the demonstration was to present a live demonstration of SATNET technology including packet speech, facsimile, and record traffic in a network including earth stations of widely different capabilities. Most of the original

goals for the demonstration were achieved, and both the demonstration and the brief live presentation at the NTC conference were judged to be successful.

Preparations for the demonstration and the NTC conference revealed several problem areas in SATNET. A large part of the SATNET frame was and still is taken up by a direct dedicated connection between London and the U.S. ARPANET. This dedicated channel reduces the total data rate available for the SATNET demand-assigned traffic; however, this is ordinarily not a problem for transmissions between the large INTELSAT earth stations. When the UET participates in the network, however, all control packets and those data packets intended for the UET must be sent as mode 2 packets (rate-1/2 coded BPSK which gives an information bit rate of 16 kbit/s). Also, the headers of all packets must be sent as coded BPSK, requiring mode 4 transmission. These transmission formats consume a larger segment of the TDMA frame to transmit a given number of information bits as compared to mode 1 (64-kbit/s) packets, and thus reduce the average capacity of SATNET. A "buddy controller" strategy, in which one large station interprets the control packets, performs the scheduling, and informs the small earth station, has been proposed to minimize this capacity loss. To date this has not been implemented in SATNET.

In preparing for the demonstration, experiments were run in a mixed mode in which packets were sent as either mode 2 (rate-1/2 coded BPSK), or mode 4 (part rate-1/2 coded BPSK, part uncoded QPSK). These modes allowed the UET to receive all or part of the packets, respectively. For mixed-mode experiments which used CPODA in the control subframe, an unexpected problem arose when collisions occurred among the control packets. Ordinarily, in this "slotted-Aloha" utilization of the control subframe, it would be expected that overlapping control packets

would be lost at all receiving stations. Attempts to use CPODA, however, showed that a surprisingly large fraction of the contending packets were acquired, demodulated, decoded, and accepted as valid by one (but not all) of the stations. When this occurs, different receiving stations have different scheduling information to work with and the frame scheduling quickly gets out of synchronization. A possible solution to this contention problem, which has been observed only for mode 2 control packets, relies on processing the T&M data and rejecting packets with abnormally low E_b/N_o values. Although this technique seems promising, it has not yet been evaluated fully in the network environment.

Because of the problems with CPODA, this method of utilizing the control subframe was abandoned prior to the demonstration, and each station was assigned fixed control slots in the control subframe (FPODA). The FPODA technique results in a slight loss of control subframe utilization efficiency as compared to that of CPODA.

The combination of these factors resulted in relatively low SATNET capacity during the demonstration. This situation required some compromises both in the demonstration configuration and in the choice of experiments to the presented at the conference session.

Following the NTC demonstration, Task 2 activities concentrated on extending demo terminal capabilities in the area of facsimile transmission. Alternative configurations for a general electronic mail capability were reviewed and a general experimental plan was developed for continuing the investigation. The hardware in the existing demo terminal will need to be upgraded to achieve some of these goals. This will not occur until the second half of 1980. The delay is due partly to the lead times for ordering certain equipment that will be ordered under a follow-on to the present contract.

Additional activities have included the successful decoding and interpretation of the data stream transmitted by the DACOM 450 facsimile terminal. A thorough understanding of this data format is necessary for archiving, editing, and format conversion (i.e., to CCITT format).

1.2.3 TASK 3: WIDEBAND SYSTEM COORDINATION/INTEGRATION

COMSAT's participation in the wideband system, under Task 3, began at an initial coordination meeting in March 1979, where information was requested from Western Union concerning transmission parameters for the wideband link. Parameters such as gain/group-delay characteristics of the earth station and satellite filters and earth station and satellite nonlinear characteristics were requested along with details of the modem filtering and modulation/demodulation technique. Frequency plans were also requested to define the exact placement of the wideband carrier in a particular transponder of one of the WESTAR satellites, and to identify the other signals that would share that particular transponder. This information was used to perform computer simulations of candidate frequency plans for the wideband carrier. Meetings were held at Western Union in April and again in July 1979, to receive the most recent planning information for the wideband communications link. This information was incorporated into the simulation model and additional simulation experiments were performed. These simulations showed only small additional losses due to channel impairments for the channel configurations that were simulated. It was concluded that the wideband link would operate as expected at the operating point of 5×10^{-3} error rate.

COMSAT's participation in the wideband domestic network continued during the fourth quarter of 1979. A coordination meeting was held at DCA in October 1979, followed by visits to Scientific-Atlanta and the Linkabit Corporation. During the visit to Linkabit, a system issue related to network timing accuracy was raised. Based on this issue, work was started on modeling the network timing functions and sources of error.

The wideband system coordination activities continued during the first quarter of 1980 and included several integration issues between Linkabit, Western Union, and BB&N. Information was also collected and transmitted to Information Sciences Institute (ISI) about the physical installation at that particular site. Work continued on the simulation model for wideband network timing. Simulation results were obtained assuming that the PSAT (Pluribus SIMP) would implement ideal second-order tracking algorithms. At this point, the simulation experiments were suspended awaiting specific information from BB&N on implementation of the tracking algorithms in the PSAT. The results obtained to date give lower bounds on the timing errors to be expected in the wideband system. Plans were made for a wideband coordination meeting at DCA, Reston, in mid-June 1980. This meeting was held June 19, 1980.

2. ACTIVITIES UNDER TASK 1: TRANSITION OF SATNET TO OPERATIONAL STATUS

2.1 INTRODUCTION

Activities under Task 1 have included experiments with the operational PSP terminals and the prototype terminal used with the UET at Clarksburg, in preparation for the transition of SATNET from experimental to operational status scheduled for mid 1980. The three PSP terminals to be installed at Etam, Goonhilly, and Tanum were developed under a previous contract with DARPA, which was completed in October 1979. The PSP terminal at Etam was installed in June and July 1979, and at Goonhilly in September and October 1979. Except for minor problems (power supply failures following equipment relocation at Etam and a broken wire on the C&M connector at Goonhilly), these two terminals have operated continuously in the SATNET network.

It was decided in the fall of 1979 to keep the third PSP terminal (scheduled for delivery to Tanum, Sweden) at COMSAT Laboratories for an extended period to support preparation for the NTC demonstration, and to allow checkout and calibration of the C&M functions, which were recently added to the terminals. These calibration measurements were completed in March 1980, and the results are documented in the third and fourth quarterly technical reports under this contract (dated February 1980 and May 1980, respectively). The terminal was shipped to Tanum in late April and installed as described in Subsection 2.2 during the first two weeks of May.

Following the NTC demonstration at the end of last year, work began on upgrading the prototype PSP terminal to include the

C&M module and various interconnections to the other subsystems. This work was completed recently, and test results over the satellite channel are included in Subsection 2.3.

During the past quarter (March 15-June 1980), initial C&M testing was conducted in conjunction with BB&N. Anomalous behavior was noted when the T&M parameters were enabled at certain earth stations. The same behavior, i.e., high missed packet rates on only one link, was noted first at Etam, then at Tanum after PSP terminal installation, and finally at Goonhilly. The cause of the behavior was isolated to an incompatibility between the format of the T&M words and the SIMP hardware. This was verified during tests at Goonhilly and later by tests made with the UET at Clarksburg.

A final activity under Task 1 was the continued monitoring of SATNET performance data. Subsection 2.4 contains summaries for April, May, and June 1980.

2.2 INSTALLATION OF THE PSP TERMINAL AT THE TANUM, SWEDEN EARTH STATION

When the PSP terminal arrived at Tanum on May 8, a visual inspection showed that no mechanical damage was experienced during shipment. The terminal was installed, power was supplied, and the IF input/output cables were connected to the IF subsystem in the SPADE terminal. Tests performed in the data and IF loopback low bands revealed that the terminal was operating normally. [The data test set (DTS) was used as the data source.] However, when tests were made in the high band*, no reception was possible.

*The SPADE transponder is divided into two bands, transmission on the low band is from 52-70 MHz, with reception from the SPADE IF subsystem at 98-116 MHz, and on the high band from 70-88 MHz, with reception at 116-134 MHz.

The problem was traced to the 45-kHz oscillator used for reference into the two synthesizers; it was off frequency by approximately 2 Hz. This caused a frequency difference of about 800 Hz between the two synthesizers, and an offset transmit frequency near 3 kHz on the high band.

The 45-kHz oscillator was removed from the terminal, and a 9-MHz signal from the NEC SCPC equipment located next to the PSP terminal was used as a signal reference. The 9-MHz signal was divided by 200 to obtain the necessary 45-kHz reference. With this new reference signal, reception on both high and low bands was achieved. Tests also showed that the terminal operated normally in the network.

Bit-error-rate (BER) measurements were made with the two modems. The carrier-to-noise ratio, as measured at Etam, was set to 17.3 dB. Using the DTS as the data source and transmitting 9999 packets with a packet length of 229 sixteen-bit words, the bit error rates for the two modems were measured as follows:

Modem A (S/N-3)	. . .	6.0×10^{-7}
Modem B (S/N-2)	. . .	6.1×10^{-5}

Because of the better BER performance of Modem A (S/N-3), the modems were switched in the rack, so that S/N-3 is located in the B-channel unit, and S/N-2 in the A-channel unit. Channel Unit B is the normal operating channel unit. The frequency offset adjustment in the modem was set to minimum with Tanum receiving its own carrier. The frequency offset from the Etam carrier was approximately 1 kHz, and from Goonhilly, approximately 500 Hz. The frequency offset measured on the pilot was approximately 100 Hz.

With the terminal operating on the test channel, tests were made to confirm C&M module operation. All commands to the channel units via SIMP commands proved satisfactory. However,

with the T&M word enabled and the terminal operating in the network, some packets received from Goonhilly were being lost; i.e., the reception of packets from Goonhilly dropped to 95 percent, whereas the reception of packets from Etam and Tanum remained close to 100 percent. This was the same phenomenon noticed at Etam when the T&M words were enabled; that is, Etam lost approximately 5 percent of its own packets while receiving Goonhilly and Tanum at 100 percent.

The Tanum PSP terminal operated in the net for two days, and the light display on the SIMP front panel indicated normal operation (close to 100-percent reception from all three stations). On the third day, a power outage momentarily interrupted system operation. The system came back up immediately and appeared to be operating normally. Monitoring of system performance continued through May 15 when COMSAT personnel left the earth station.

After leaving Tanum, COMSAT Laboratories personnel visited Goonhilly to perform tests with the C&M module. Reports from BB&N showed that the T&M parameters could not be turned on via a SIMP command. Investigation showed that no command strobe signal was being received by the Linkabit interface. Other C&M commands could be activated, for example, data loopback and DTS commands. It was determined that the wire carrying the command signal from the C&M module to the Linkabit interface had either fallen off the backplane connector or had not been installed during fabrication. The connection was made to the backplane and the T&M words could now be enabled. With this correction, the PSP terminal at Goonhilly was completely operational.*

*The Phase I interface units have not yet been installed in the Goonhilly and Tanum PSP terminals. This will be done when the SPADE channel units are restored. The transmit and receive interface units presently used in SPADE will be converted for use in these terminals.

The same T&M tests were performed at Goonhilly after the PSP terminal backplane was repaired. When the T&M words were enabled at Goonhilly, the same phenomenon was again observed; however, Goonhilly received only Etam at 100 percent, whereas reception of hello packets from Goonhilly and Tanum was reduced to 95 percent. At all three stations, prior to the enabling of the T&M words, reception was 100 percent. Afterward, burst reception from one or two stations dropped to 95 percent; disabling the T&M words restored reception to 100 percent.

To verify that the structure of the T&M words causes the missed packets, an experiment was performed at Goonhilly using the following procedure:

a. With the T&M words disabled, sufficient time was allowed for the SIMP front panel lamps to register 777, indicating 100-percent reception from all sites.
b. The T&M word was then enabled, and sufficient time was allowed for the lamp display to drop to 7**xx** (**x** = 3 or 4).
c. The T&M words from the modem were then forced to be all zeros. The lamp display returned to 777, even though the T&M words were still enabled.
d. Items b and c were repeated. The lamp display again dropped to 7xx with T&M words normal, and 777 with T&M words all zero.

It appears that when T&M words are enabled, at least one byte of data contains the DLE data sequence (020 octal), which is detected by the Honeywell 316 interface. If the DLE is not followed immediately by an ETX data sequence (203 octal), the byte is discarded by the SIMP software; however, the software continues inputting data. When the true DLE ETX is detected by the Honeywell 316 interface, the SIMP software then examines the

total packet for proper length. Since the packet does not contain (N + 4) 16-bit words between SYN DLE STX and DLE ETX (one byte was discarded after DLE detection in the T&M words), the entire packet is discarded. The T&M byte most likely to generate the DLE sequence is the byte which estimates frequency offset. A frequency offset of +250 ± 7.8 Hz corresponds to 020 octal in the T&M word. A frequency offset of this magnitude is not unreasonable or uncommon.

There are three possible solutions to this problem. The first is to have Linkabit restructure the PROMS for all of the modems so that the first bit of each byte is "1" rather than "0". This will ensure that the DLE sequence will not appear in the three T&M words generated by the modem. Also, the CRC bit in the fourth T&M word generated by the Linkabit receive interface would have to be inverted from its present value, i.e., changed to CRC GOOD = 1, BAD = 0.

The second solution would require BB&N to reconfigure the hardware and software in each SIMP so that it does not search for a DLE ETX until after the T&M words have been received. This would require modifications to both the hardware and software.

The third solution would be to have the COMSAT receive interface (SSIC Rx) invert only the T&M words transferred between the modem and the Linkabit interface, SSIL Rx. This would require a relatively simple modification of the existing interface hardware. This does not guarantee that the fourth T&M word would not contain the DLE data sequence, however, and this fourth word can create problems for transmission modes using BPSK. In these modes, the 4-bit word indicating disagreement in the correlation of the BPSK SOM sequence takes on specific (relatively low) values. One value (2) will create "DLE" in the second byte of the fourth T&M word. This will cause a problem primarily at the UET where two disagreements are expected about 5 percent of the time.

To summarize, the "quick fix" described above should solve the T&M problem at the three large earth stations. This problem may remain at the UET during the reception of BPSK transmissions (i.e., Mode 2 or 4 when the network is operated in mixed mode).

After the 45-kHz oscillator from the Tanum PSP terminal arived at COMSAT Laboratories about 2 weeks after the installation, it was tested and found to be off frequency by approximately 2 Hz. The oscillator was returned to the manufacturer for repairs. After repairs were completed, the oscillator was checked at COMSAT Laboratories before it was shipped back to Tanum for reinstallation in the PSP terminal. Figure 2-1 shows the output spectrum of the oscillator. It can be seen that any phase noise is almost 80 dB below the main lobe for frequencies further removed from the carrier than 80 Hz. The manufacturer also added a tuning capacitor so that the oscillator output can be set exactly to frequency.

2.3 UET TERMINAL MODIFICATION

The prototype PSP terminal used with the UET at COMSAT Laboratories, Clarksburg, Md., was modified during the second quarter of 1980. The modifications included the addition of the C&M module and a SSIC Rx identical to the ones used in the PSP terminal at the standard earth stations. These changes required that the entire backplane of the terminal be refabricated and rewired to accommodate the extra connectors required by the various modules. New switch matrix and end-of-packet generator modules were also needed. In addition, the DTS, which was returned from Linkabit, had to be rewired to work with the C&M module. This task was completed near the end of the contract period. Tests performed through the satellite with both the DTS and the SIMP were successful.

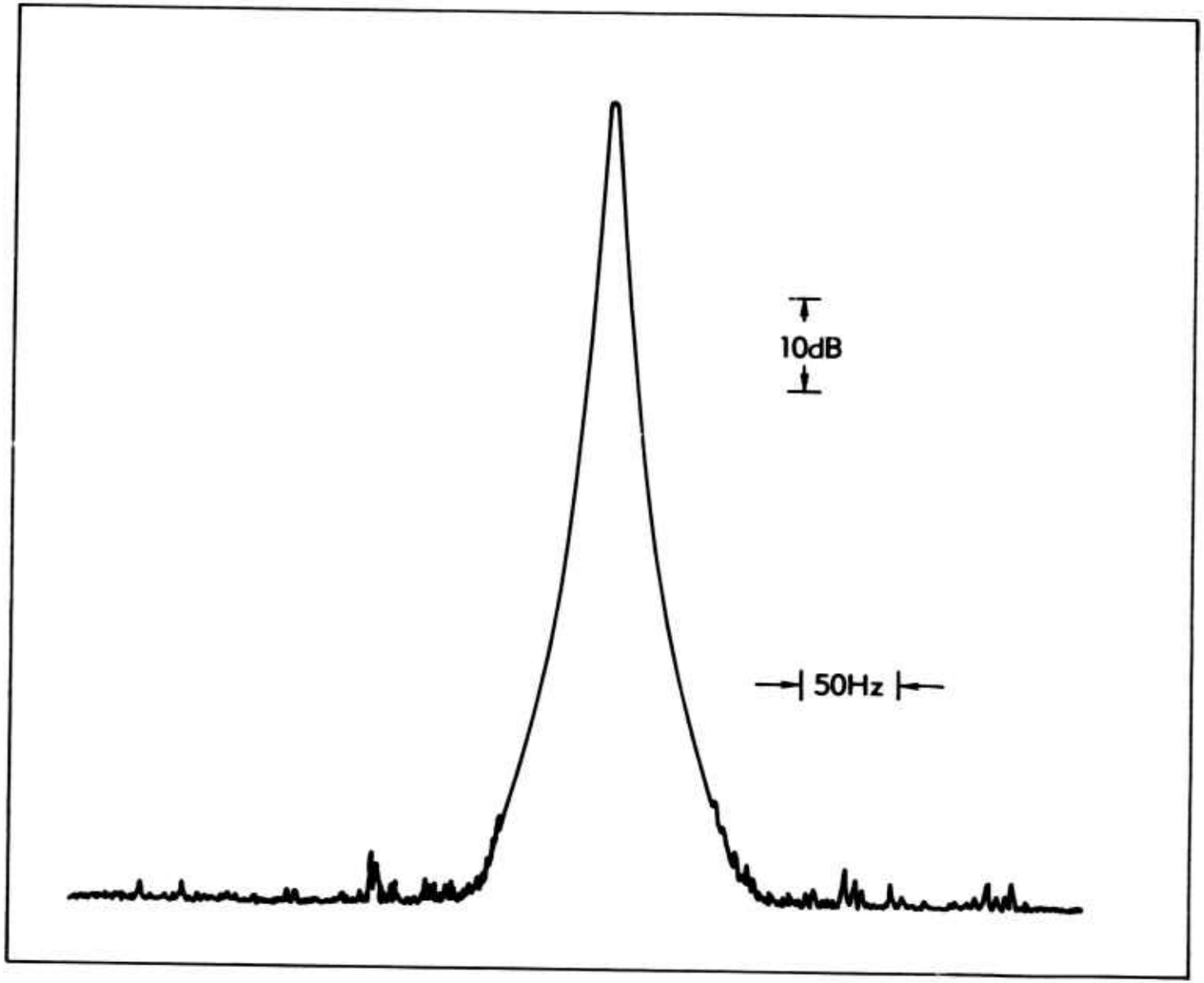


Figure 2-1. 45-kHz Oscillator Output Spectrum

Additional tests were made with the UET operating in SATNET with Etam and Goonhilly.

The basic link budget relationships for the SPADE transponder used in SATNET are as follows:

$(C/N_o)_{UP}$: $68.3 - L_u$ (dB-Hz)
$(C/N_o)_{IM}$: 69.0 (dB-Hz) assumed worst case
$(C/N_o)_{DN}$: $64.2 - L_u - L_D - 40.7 + (G/T)_{ES}$ (dB-Hz)

where

L_u and L_D = abnormal up- and down-link losses (for example, rain loss or losses due to antenna pointing errors) which are assumed to equal 0

$(G/T)_{ES}$ = gain-to-noise temperature ratio of the receiving earth station (the UET in this case which has a G/T ratio of 28.5 dB/K).

The combined values for $(C/N_o)_T$, i.e., the reciprocal of the sum of the reciprocals of the three contributors, yield overall $(C/N_o)_T$ values of 61.8 dB-Hz into a large, Standard A station. If intermodulation noise is neglected, $(C/N_o)_T$ increases to 62.7 dB-Hz. For a noise bandwidth of 38 kHz, these two values yield C/N ratios of 16 dB (worst-case intermodulation) and 17 dB (neglecting the intermodulation). An intermediate value of C/N = 16.5 dB* (in a 38-kHz bandwidth) appears to be a nominal operating condition for SPADE carriers received at Standard A earth stations. For 64-kbit/s operation, the nominal operating E_b/N_o value for the large stations is 13.7-14.6 dB; the lower number assumes worst-case intermodulation noise.

*Much of the earlier SATNET work was based on this reference value being 17.5 dB. The lower value (16.5) is correct as of mid 1980--a change was evidently made by INTELSAT early this year.

Using the nominal value of 28.5 dB/K for the G/T ratio of the UET, a down-link C/N_o ratio of 52.0 dB-Hz can be obtained. The UET is almost completely "down-link limited" with overall $(C/N_o)_T$ values predicted in the range 51.8-51.9 dB-Hz (with and without worst-case intermodulation noise, respectively). In a 38-kHz bandwidth, the C/N ratio at the UET is approximately 6.0 dB. For reception at information bit rates of 64, 32, and 16 kbit/s, the nominal E_b/N_o ratios would be 3.8, 6.8, and 9.8 dB, respectively, at the UET.

Figure 2-2 shows measured bit-error rate versus energy per information bit-to-noise density ratio (E_b/N_o) for the UET modem/system for transmission of Mode 2 packets (BPSK-coded rate 1/2 giving an information bit rate of 16 kbit/s). At the nominal operating point of E_b/N_o = 9.8 dB, the BER is considerably less than 10^{-7} and cannot be measured. The figure also shows BER versus E_b/N_o for uncoded QPSK (Mode 1), and the theoretical curve of BER vs E_b/N_o. Figure 2-3 shows the missed packet rate vs E_b/N_o for Mode 2 packets. At the nominal operating point, the missed packet rate is approximately 7×10^{-4} (99.93 percent receptions).

Bit-error-rate measurements were also performed with Mode 3 packets, which are QPSK-coded rate 1/2 (information bit rate of 32 kbit/s). Mode 3A has a short, 64-symbol, preamble and Mode 3B has a long, 96-symbol, preamble. The nominal operating point for both Modes 3A and 3B is at E_b/N_o = 6.8 dB. Figure 2-4 shows the results of these tests. Figure 2-5 shows the bit-error rate for Mode 3B when operating on the test channel through the satellite. The results compare favorably with the back-to-back tests in Figure 2-4.

Along with the high bit-error rate (3×10^{-3}) for Mode 3 operation at nominal operating points, Figure 2-3 indicates that 20-25 percent of the packets are "missed". The missed packet rate is high at the nominal operating point because the quadrature SOM

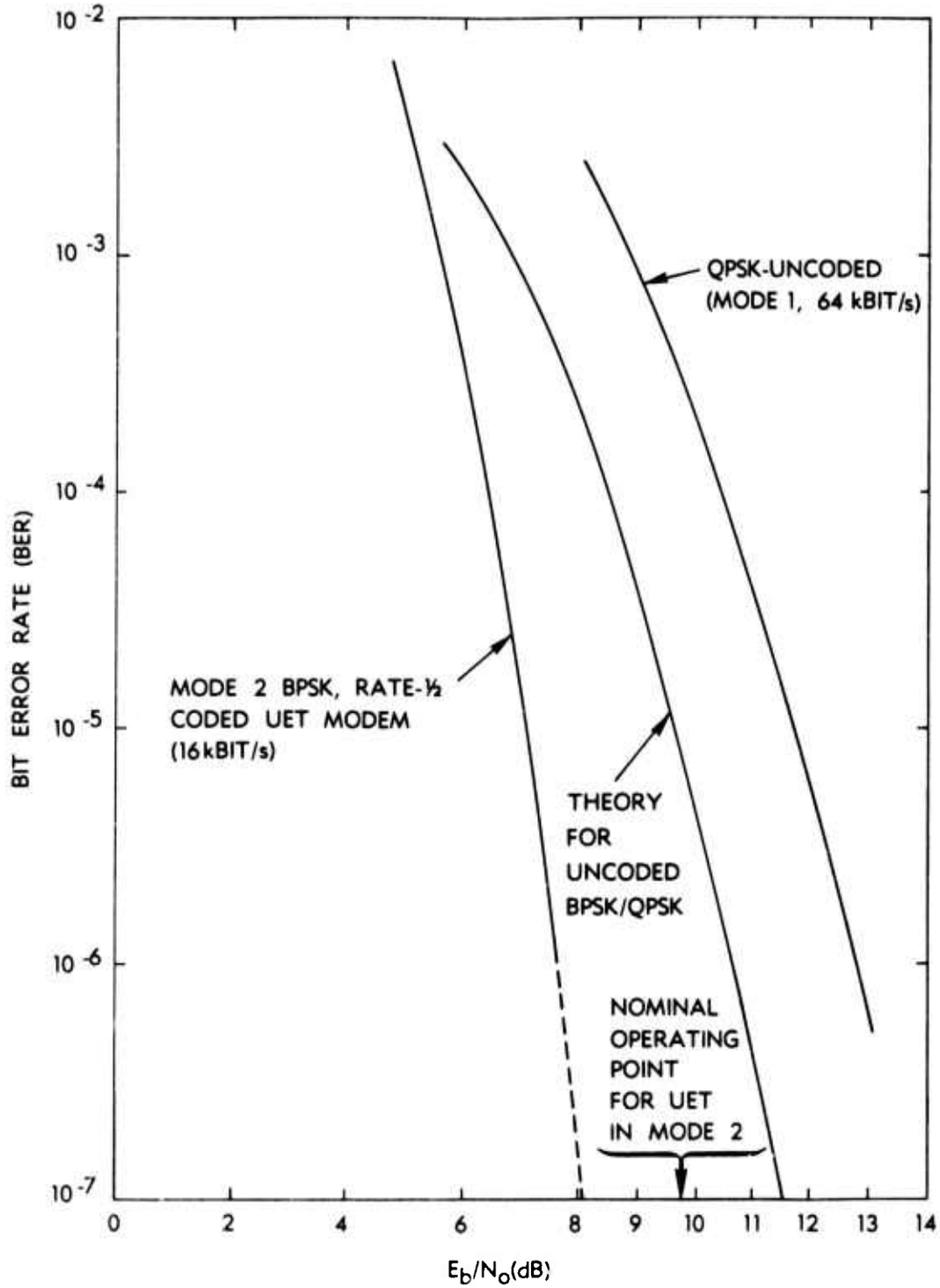


Figure 2-2. BER vs E_b/N_o--UET Modem in Prototype PSP Terminal

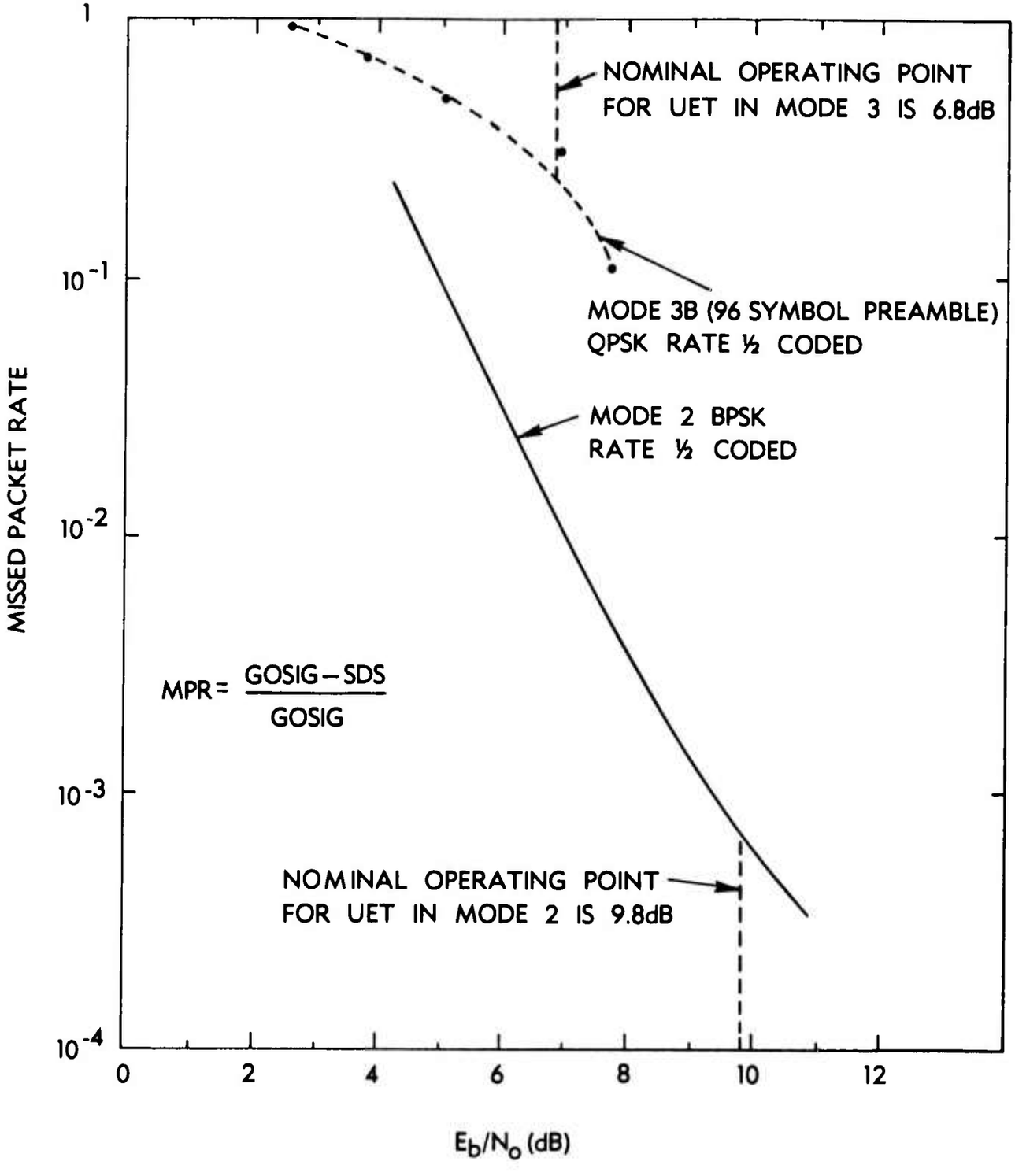


Figure 2-3. Missed Packet Rate vs E_b/N_o

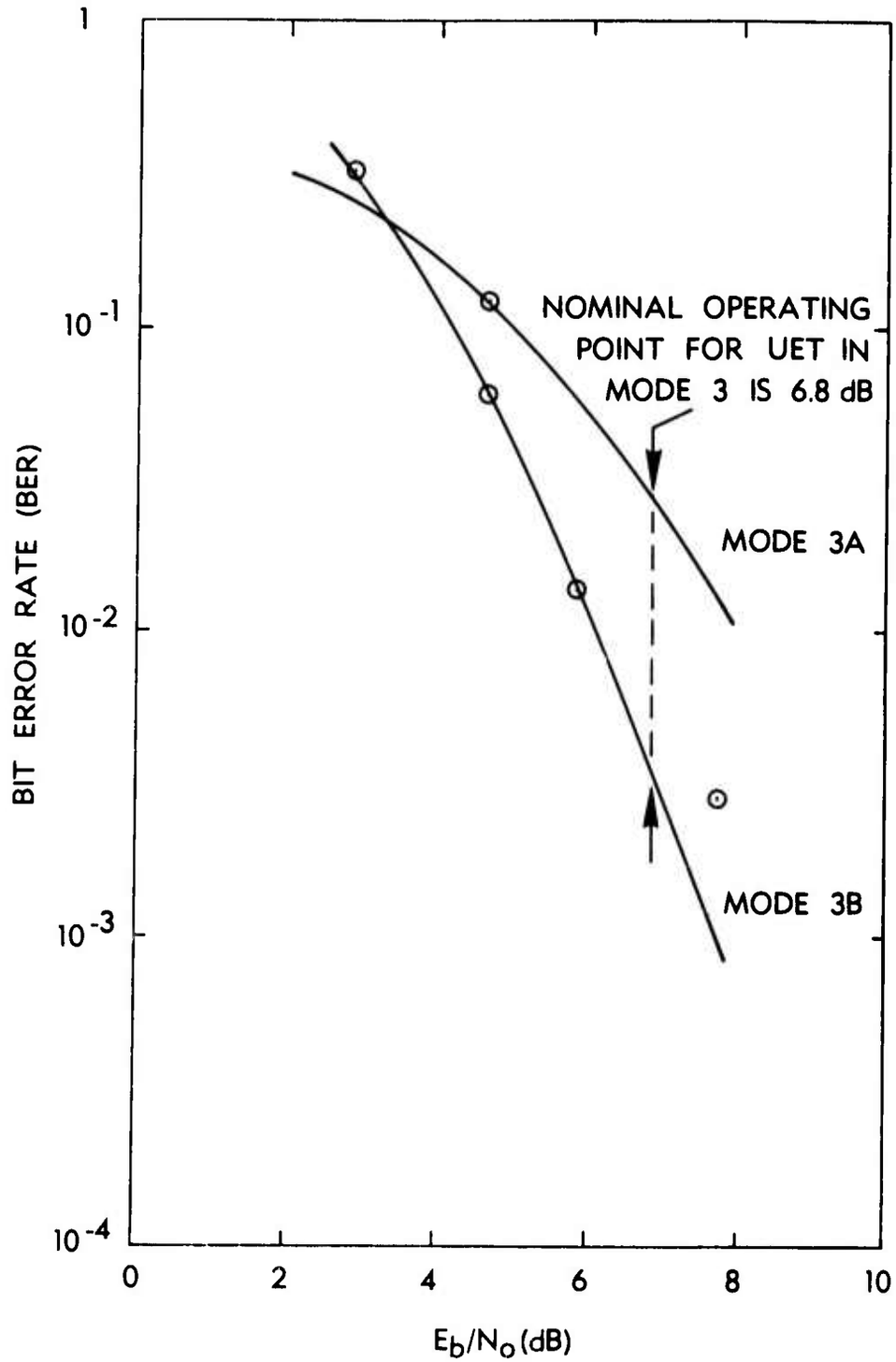


Figure 2-4. BER vs E_b/N_o Packet Modes 3A and 3B

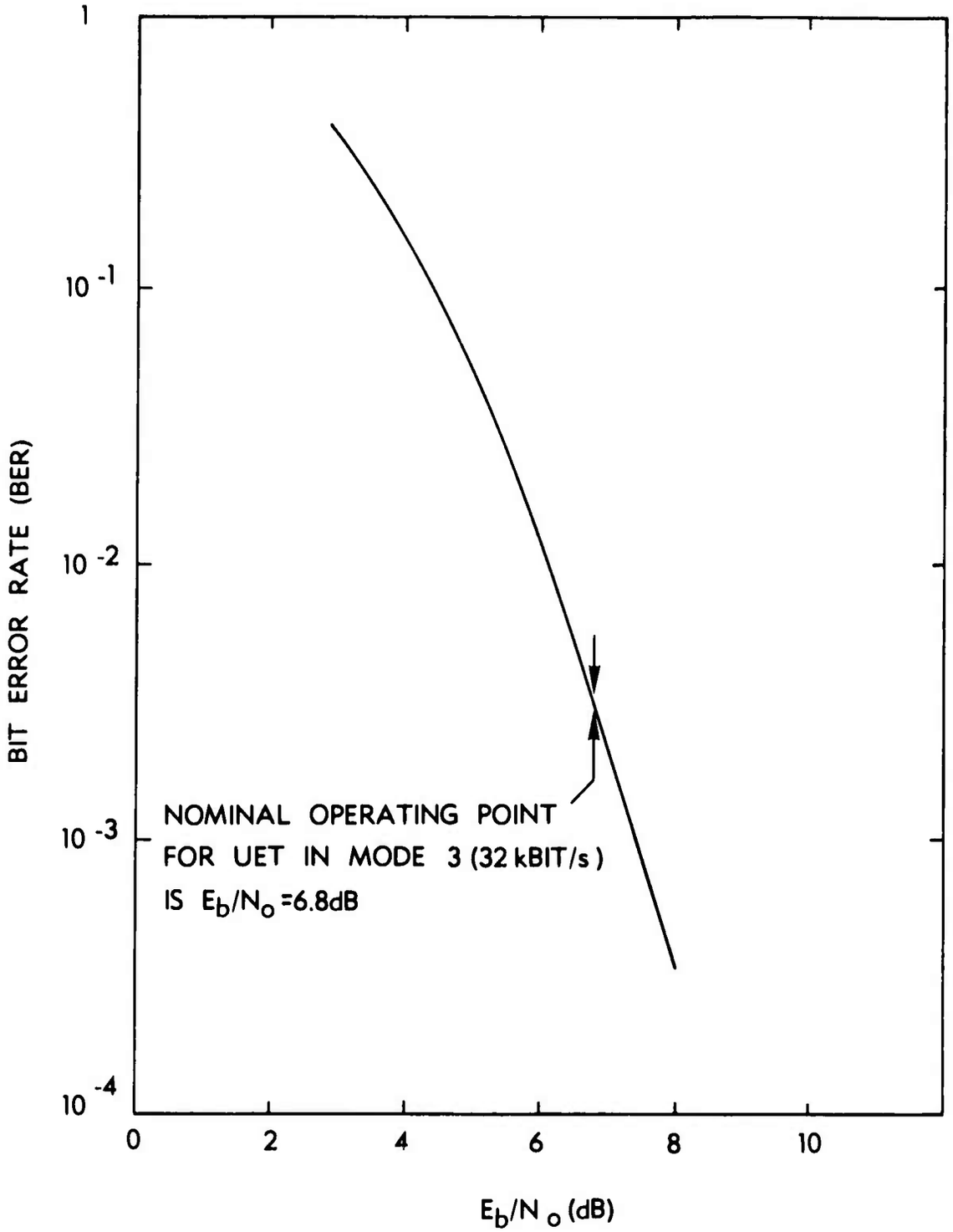


Figure 2-5. Bit Error Rate (BER) vs E_b/N_o

sequences (unique words) used to resolve phase ambiguities are only 16 bits long for each channel, and these words are being received at a very high "raw" channel error rate. For operation at an E_b/N_o of 6.8 dB with rate 1/2 coding, the binary elements comprising the in-phase and quadrature unique word sequences are being received at energy contrast ratios of 3 dB less, or 3.8 dB. The raw channel error rate on these elements is approximately 0.02, and possibly a little higher due to modem implementation losses.

The probability that the unique word is missed (three or more disagreements in either or both of the 16-bit unique words on the I and Q channels) can be approximated as*

$$\Pr\{{}^{\text{SOM}}_{\text{MISSED}}\} \tilde{\approx} 496\ p^2 + 11040\ p^3 + 96960\ p^4$$

where p is the raw channel error rate. Assuming p = 0.02, this expression gives a miss probability of 0.3 if the threshold is set at two disagreements. This value agrees roughly with the observed miss performance in Figure 2-3. Increasing the number of allowable bit errors in the SOM from two to three (i.e., decreasing the detection threshold) increased the number of packets received (i.e., SDS detection) by a few percentage points as shown in Table 2-1.

In addition to the problem with SOM detection, which appears to account for the missed packet behavior, the Linkabit interface must also detect and correctly interpret the 24 bits of SYN DLE STX sequence and the 32 bits (2 words) of header information. At the nominal operating point, although these data are coded, they are being received at a decoded bit-error rate that

*Final Report, "COMSAT participation in ARPA Packet Satellite Program," Contract No. F04701-C-0240, COMSAT Labs, October 1979, pp. 6-7.

is relatively high, 3×10^{-3}. Thus, both the SDS and header sequences would be expected to contain bit errors a relatively high percentage of the time, possibly 5 percent each. These errors would cause erratic packet detection (and they cause problems in making bit-error-rate measurements with the existing test set). Thus, although SOM detection could be improved by a redesign to add longer unique words, this single step will not solve completely the missed packet problem in mode 3. The link conditions are such that the UET is approximately 3 dB short of achieving reliable mode-3 operation.

Table 2-1. Percentage of Packets Received vs E_b/N_o

E_b/N_o	Percent Reception	
	Two-Bit Errors	Three-Bit Errors
2.6	5.1	6.4
4.8	46.6	49.0
7.1	82.6	84.9

2.4 SUMMARY OF SATNET PERFORMANCE DATA

Tables 2-2, 2-3, and 2-4 summarize SATNET performance for April, May, and June 1980. The nine links are identified as follows (E = Etam, G = Goonhilly, T = Tanum):

		Receiving		
		E	G	T
Transmitting	E	1	2	3
	G	4	5	6
	T	7	8	9

Table 2-2. Missed Hello Packets--April 1980

Day	Missed Hello Packets: Links 1	2	3	4	5	6	7	8	9	Daily Average	Outage Correction Factor	E*	G*	T*
1	61	4	7	1	3	3	23	12	5	13.2	1.03	43	3	2
2	43	5	0	0	3	2	17	78	11	17.7	-	22	2	0
3	30	3	3	11	1	5	43	332	18	49.6	-	13	1	0
4	1	0	0	0	0	0	28	97	4	14.4	6.64	0	0	0
5	25	6	6	0	1	36	55	479	36	71.6	-	18	0	0
6	48	14	3	4	0	42	16	551	34	79.1	-	40	6	2
7	85	17	7	3	5	30	22	189	16	41.6	1.15	61	7	0
8	25	6	8	0	3	3	40	223	17	36.1	-	19	1	3
9	32	11	20	5	2	24	64	393	92	71.4	-	36	6	4
10	14	9	1	0	0	8	20	75	9	15.1	2.41	11	0	0
11	34	73	29	1	17	31	28	344	24	64.6	-	29	3	1
12	41	19	8	2	0	19	22	59	27	21.9	-	38	10	1
13	60	22	6	1	0	19	28	210	27	41.4	-	46	4	3
14	58	24	23	17	43	92	10	440	100	89.7	-	54	35	0
15	402	661	1319	91	103	36	15	1414	25	451.8	-	378	138	13
16	746	319	190	56	198	102	75	257	15	217.6	1.04	636	385	17
17	4	52	1	0	2	1	4	2527	3	288.2	8.42	36	1	0
18	349	509	404	137	65	12	43	2182	40	415.7	-	583	56	4
19	60	75	23	14	14	15	31	3201	45	386.4	-	62	22	0
20	13	7	1	2	0	3	5	406	6	49.2	5.37	12	1	0
21	41	12	5	34	8	21	25	105	11	29.1	1.18	30	39	7
22	118	16	19	102	17	17	67	15	24	43.9	-	81	81	30
23	185	37	17	221	96	66	234	79	69	111.6	1.03	126	190	129
24	74	11	9	69	13	4	66	9	17	30.2	-	51	60	31
25	94	16	4	97	7	7	80	25	13	38.1	-	48	75	39
26	89	23	7	53	5	9	38	23	15	29.1	-	60	49	20
27	149	35	10	132	35	24	244	169	152	105.6	1.04	93	98	75
28	84	36	36	81	16	20	66	74	46	51.0	-	69	79	46
29	14	9	9	26	11	5	16	13	11	12.7	-	13	32	10
30	29	5	7	57	4	9	45	9	13	19.8	-	20	48	17
Monthly Average	100	68	73	41	22	22	49	466	31	96.9	N/A	91	48	15

*Total hardware checksum errors.

Table 2-3. Missed Hello Packets--May 1980

Day	Missed Hello Packets: Links 1	2	3	4	5	6	7	8	9	Daily Average	Outage Correction Factor	E*	G*	T*
1	59	4	3	63	3	4	92	11	12	27.9	1.57	37	39	47
2	77	4	41	70	27	86	155	91	92	71.4	1.04	24	36	43
3	62	4	6	60	7	7	99	14	13	30.2	-	37	36	48
4	54	5	3	69	5	4	96	12	13	29.0	-	30	43	41
5	44	13	28	34	11	2	79	11	17	26.6	-	55	28	33
6	60	19	11	50	46	2	200	39	24	50.1	1.11	54	64	52
7	65	18	18	71	26	5	95	18	23	37.7	-	56	69	36
8	52	25	67	71	9	2	79	15	20	37.8	-	74	49	35
9	5	5	20	9	10	16	8	9	8	10.0	-	15	15	3
10	29	9	57	24	8	49	80	84	81	46.8	-	29	24	8
11	21	21	29	23	7	17	13	8	11	16.7	-	58	25	4
12	24	10	40	22	10	19	52	49	46	30.2	-	48	28	7
13	58	64	126	81	94	167	153	508	602	209.2	-	105	134	658
14	12	13	19	15	24	111	123	287	338	104.7	1.69	22	71	366
15	857	196	11	8	0	82	106	4	14	142.0	2.47	113	12	3
16	50	12	29	55	72	212	396	788	1223	315.2	-	49	237	1020
17	37	14	407	24	17	540	812	1472	2920	693.7	-	219	305	1896
18	34	18	1067	72	80	1222	492	1153	2565	744.8	-	527	674	1679
19	22	44	1108	274	223	1434	275	609	2058	671.9	-	560	918	1550
20	62	173	825	189	301	1177	112	783	1044	518.4	-	430	659	867
21	66	99	363	213	283	684	148	1352	768	441.8	-	232	485	969
22	4	22	499	279	305	731	48	462	2208	506.4	-	242	597	777
23	4	6	73	192	231	230	219	662	2814	492.3	-	40	336	1015
24	165	33	130	0	1	8	18	28	44	47.4	-	161	4	8
25	29	17	315	83	151	377	278	739	1963	439.1	-	165	357	1173
26	27	21	412	88	-	506	85	251	557	243.4	-	247	531	523
27	464	485	1844	503	551	1824	1060	3496	7341	1952.0	-	191	423	3448
28	14	5	445	166	25	560	706	4122	5927	1330.0	-	184	398	3723
29	107	17	17	3	37	7	10	27	10	26.1	-	102	39	15
30	14	3	62	150	36	117	322	2071	2514	586.7	-	38	196	1534
31	41	-	11	112	27	40	102	489	664	165.1	-	38	92	155
Monthly Average	85	46	261	99	88	330	210	635	1159	324	N/A	135	223	701

*Hello packets received with hardware checksum errors.

Table 2-4. Missed Hello Packets--June 1980

Day	Missed Hello Packets: Links 1	2	3	4	5	6	7	8	9	Daily Average	Outage Correction Factor	E*	G*	T*
1							-				-		-	
2	16	2	52	79	30	82	161	403	391	135.1	1.01	18	109	96
3	194	66	96	149	235	121	368	675	1,398	366.9	-	134	315	286
4	5	36	76	79	285	84	225	545	1,591	325.1	-	63	295	186
5	99	83	123	46	50	97	97	-	259	106.8	1.06	79	79	114
6	67	43	145	98	69	201	248	694	956	280.1	-	87	136	227
7	69	6	1,592	258	37	3,369	327	1,710	2,814	1,131.3	-	956	2,105	946
8	33	5	437	139	33	852	313	1,240	1,506	506.4	-	220	505	416
9							-				-		-	
10	27	4	271	108	30	798	477	927	1,177	424.3	1.16	135	518	406
11	54	23	2,437	117	35	3,189	1,887	2,407	4,119	1,585.3	-	1,703	1,567	1,732
12	14	0	15	43	18	16	88	177	78	49.9	2.87	16	62	85
13	79	72	59	137	49	54	173	370	-	124.1	-	59	155	168
14	78	26	2,096	142	44	2,174	2,428	2,207	2,783	1,330.9	-	1,730	1,730	1,851
15	297	10	20	178	40	49	191	593	176	172.7	-	126	188	155
16	279	208	108	381	290	196	1,936	9,606	897	1,544.6	-	183	422	290
17	2	1	3	8	3	26	111	1,776	34	218.2	6.40	3	19	16
18	5	110	57	58	189	138	782	8,097	279	1,079.4	-	49	193	206
19	4	0	7	2	1	7	183	3,097	62	373.7	2.98	6	5	53
20	12	138	44	9	63	66	524	6,788	204	872.0	-	40	80	183
21	3	3	37	0	1	36	399	14,938	195	1,734.7	-	17	12	165
22	12	-	-	3	0	51	488	15,945	141	2,377.1	-	21	17	149
23	30	18	86	10	6	111	581	21,162	240	2,471.6	-	58	42	177
24	7	26	33	1	5	48	406	11,331	153	1,334.4	1.43	38	19	151
25	537	10	54	3	4	54	777	8,610	341	1,154.4	-	261	38	307
26							-				-		-	
27	27	317	42	7	127	40	353	16,434	156	1,944.8	-	346	116	132
28	24	84	32	9	4	27	841	17,288	347	2,072.9	-	83	15	177
29	31	15	43	29	8	45	606	8,317	286	1,042.2	-	37	33	104
30	323	298	366	351	315	348	961	2,945	594	722.3	-	180	229	333
Monthly Average	86	62	320	91	73	455	590	6,088	815	943.7	N/A	246	333	337

*Total hardware checksum errors.

3. ACTIVITIES UNDER TASK 2: INTERNETWORKING EXPERIMENTS

3.1 INTRODUCTION

During the first half of the contract period, activities under Task 2 focused on the NTC demonstration. This milestone is summarized in Subsection 3.2. Since the NTC demonstration, work has continued on developing the hardware and software capabilities of the demo terminal, which is located at COMSAT Headquarters, in Washington, D.C. Subsection 3.3 outlines concepts for internetworking experiments, and Subsection 3.4 gives details related to DACOM 450 facsimile decoding. The results of delay measurements made in the internet environment are summarized in Subsection 3.5, and hardware and software developments related to the internetting activity are documented in Subsection 3.6. Appendices A and B contain software details.

3.2 NTC DEMONSTRATION

3.2.1 BACKGROUND

The NTC demonstration was originated in November 1978 as a vehicle to demonstrate the capabilities of SATNET and assess its performance. Two aspects of SATNET technology were believed to have maximal impact in a live demonstration: the capability of SATNET to handle simultaneous transmission of perishable data such as live speech along with other packet data traffic, and the capability to operate effectively in mixed-rate systems including earth terminals of disparate performance.

During the planning phase, it was considered desirable to highlight those aspects of the SATNET design which advance the state of the art vis-a-vis current systems. Probably the most important is the integration of real-time speech, which must be delivered subject to critical delay limitations, and record data, which can be delayed for efficient channel utilization. The equipment installed at the various experiment sites provided the capability to support real-time speech and record data between domestic U.S. gateways and University College London (UCL), and between these sites and the U.S. and European ARPANET. As an example of record data, a facsimile capability was available and work proceeded on implementing appropriate interface software. Although the demonstration probably could have been conducted without the facsimile capability, real-time speech was considered necessary and hence was given first priority.

Another important goal was the capability to demonstrate full SATNET connectivity via the Clarksburg gateway and the UET; however, the present SIMP software can provide only marginal support because of two factors. First, a special temporary connection had to be made via SATNET to support older protocols until they could be phased out. Second, control information coordinating the channel scheduling process is sent at the rate of the lowest-rate earth station in the network, 16 kbit/s when Clarksburg is active. Accordingly, at least for the demonstration, Clarksburg could not be used for speech.

The scenario planned to verify the live-speech capability was simultaneous operation of a linear predictive vocoder, built by Lincoln Laboratories and called the LPCM, and a commercially available DACOM 450 digital facsimile machine. The LPCM data represented the perishable data, because playback delayed more than an established maximum delay would be useless. Conversely, the DACOM facsimile data represented an example of

data which could be queued and delayed for relatively long times. For the demonstration, both of these devices were operated at 2400 bit/s, although transfer of prerecorded facsimile data from the DACOM facsimile machine occurred at substantially higher rates.

The scenario designed to verify the capability of operating in mixed-rate modes involved the UET at Clarksburg, and two of the three standard INTELSAT stations in SATNET, namely, Etam and Goonhilly (Tanum was not equipped to operate at mixed rates). In the demonstration, the Clarksburg station at COMSAT Laboratories could receive at 16 kbit/s, and the two large earth stations could receive at both 16 and 64 kbit/s. Therefore, all traffic sent to Clarksburg, including packets addressed to the station and the header portion of all packets, was transmitted at 16 kbit/s. Only the data portion of the packets sent to the two large earth stations could be sent at 64 kbit/s.

In addition to demonstrating the usefulness of SATNET, the NTC demonstration illustrated the mechanisms for SATNET monitoring and control, as well as its connections to neighboring networks on both sides of the Atlantic. The demo terminal, a small computer system, allowed all of these functions to be integrated, managed, and displayed. Additional equipment for presentation aids included a number of terminals connected to the demo terminal as a host as well as to other hosts on the ARPANET, and a G.E. light valve, which was used to project selected terminal displays on a large screen.

The demo terminal includes interfaces for the Lincoln Laboratories LPCM vocoder and DACOM 450 facsimile scanner/printer, and a number of peripheral devices. The software for this system is compatible with recent internet protocols developed within the ARPANET community, including TCP-4, TELNET, and special protocols developed for real-time speech.

For the demonstrations and experiments, the demo terminal was connected to a PDP-11 gateway computer at COMSAT Laboratories. This machine, which functions as an interface to SATNET, is connected to the Clarksburg SIMP in a manner similar to other gateways, including those at BBN (Etam), NDRE (Tanum), and UCL (Goonhilly). However, at Clarksburg, the earth station of receive only at a 16-kbit/s rate, rather than the 64-kbit/s rate used by the other stations. This complicates the channel scheduling algorithm used to determine which station transmits at a particular time, and the software to support this mixed-rate operation is not yet completely developed.

The overseas partner for the demonstration was UCL, which had a similar complement of LPCM, DACOM, and demo terminal equipment. Their configuration was more complicated since the equipment was shared with other activities and was only a part of a large collection of hosts, networks, and gateways. A number of hosts on the ARPANET were also used for support, including machines at Bolt, Beranek, and Newman (BBN) in Cambridge, Massachusetts, and Information Sciences Institute (ISL) in Marina del Ray, California.

3.2.2 SYSTEM CONFIGURATION

Figure 3-1 shows the detailed system configuration. The apparent complexity is due to several factors, including provision of backup paths, coordination between the various participants, and requirements for real-time monitoring and configuration control. The demonstration site, a meeting room in the Shoreham Hotel in downtown Washington, D.C., contained the demo terminal, LPCM and DACOM devices, utility terminals, and dial-up and dedicated data circuits.

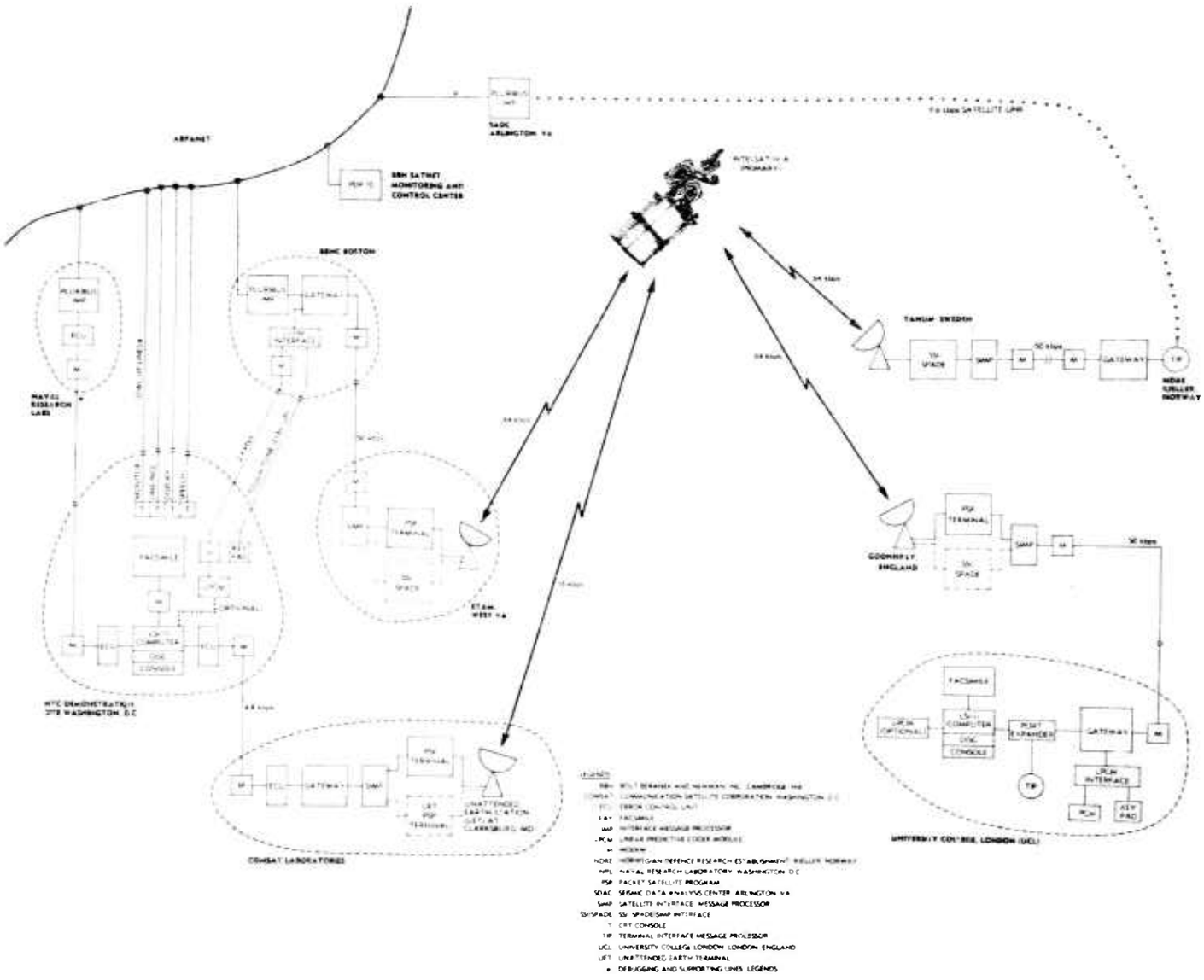


Figure 3-1. Network Configuration for Demonstration

Of the three dedicated data circuits, one 2400-bit/s circuit terminated at BBN, another 4800-bit/s circuit terminated at COMSAT Laboratories, and the third at the Naval Research Laboratories (NRL) in Washington, D.C. The first was used for transmitting LPCM data to the BBN Gateway, which connects to SATNET via a line to the Etam SIMP. The second was used to connect the demo terminal to the Clarksburg SIMP and the UET via the COMSAT Gateway, whereas the third was used as a backup path for demo terminal connection to the ARPANET via NRL. This path would be used to send traffic to SATNET via the ARPANET and the BBN Gateway if the Clarksburg station became inoperable.

Of the five dial-up circuits, one was used for voice coordination with Lincoln Laboratories and BBN during equipment setup and checkout. The remaining four were used with 300- and 1200-bit/s data sets for access to nearby ARPANET TIPs; for coordination with UCL, Lincoln Laboratories; for real-time status reporting from SATNET monitor programs on BBN and ISI hosts; and to operate various control programs for speech, facsimile, and utility file-transfer.

The two 4800-bit/s dedicated data circuits were operated on an ARPANET 1822 transparent basis. This is the normal protocol for connecting a host to an ARPANET IMP or TIP and is normally used over distances not exceeding hundreds of feet. To provide connections on this basis over the distances required, a set of error control units (ECUs) were required for each circuit. These devices, manufactured by Associated Computer Consultants under DARPA contract, provide for 1822 bit-serial operation with self-contained retransmission and supervision features using standard data sets. The four ECUs, one at each end of the two data circuits, operate with an HDLC line protocol with messages of lengths to the ARPANET maximum of about 8,000 bits.

The configuration at Clarksburg consisted of the COMSAT Gateway connected to the Clarksburg SIMP and UET. The interface between the SIMP and radio-frequency circuitry of the UET, the PSP terminal, is a highly sophisticated collection of microprocessors, modems, and codecs specially designed for SATNET service. The PSP terminal can operate in several mixed-rate modes and formats and uses state-of-the-art burst modems built by Linkabit. PSP terminals of identical design were also used at Etam and Goonhilly, and a terminal was subsequently installed at Tanum, Sweden (see Figure 3-1).

The configuration at UCL consisted of a demo terminal, a linear predictive coder module (LPCM) and DACOM devices nearly identical to those in Washington. However, the UCL Gateway is colocated with other equipment which simplifies the configuration, since no special data circuit was needed to connect the LPCM to the gateway. The port expander and TIP shown in Figure 3-1 were necessary in certain configurations during setup and checkout, but were not used in the demonstration when the demo terminal was connected directly to the gateway.

One key issue during demonstration planning was whether to connect the LPCM via the demo terminal or directly to a gateway. The reasons for the decision to connect directly to the BBN Gateway are complex and involve numerous tangential issues; however, the principal consideration was that software was available to service the chosen configuration. Furthermore, this configuration avoided the buffering and possible retransmission delays of the ECUs on the other paths. Also, it used only the high-speed (64-kbit/s) SATNET stations, providing the best performance achievable.

3.2.3 CONDUCT OF THE DEMONSTRATION

Besides demonstrating the intrinsic worth and performance of SATNET, it was necessary to monitor and control the network during the potentially disastrous disruptions that occurred in supporting systems. For instance, on the morning of the conference session, all of the ISI supporting hosts were disabled by a motor-generator failure. Transmitter power adjustments were being made at both Etam and Goonhilly until just before the conference session; this rendered Clarksburg inoperable. In addition, disruptions apparently due to faulty equipment at BBN forced all monitoring and control functions to be switched to ISI, which further complicated the restoral of operation.

Coordination at UCL was especially difficult, since no alternate path other than SATNET was available. Until shortly before the demonstration, a path from UCL to the U.S. ARPANET was available via a 9.6-bit/s line to the NDRE TIP at Kjeller (see Figure 3-1) and the NORSAR satellite link to the SDAC Pluribus IMP at Arlington, Virginia. This link, which was separate from SATNET, was available only as far as Kjeller. However, for other reasons, a special facility has been built into the SATNET control programs in the SIMPs. That is, a portion of the SATNET channel capacity has been provided in the form of a full-duplex dedicated portion of the TDMA frame connecting the London TIP at UCL and the U.S. ARPANET via the SADC Pluribus IMP. This connection uses lines, not shown in Figure 3-1, between the London TIP and the Goonhilly SIMP, and between the Etam SIMP and the SADC Pluribus IMP. Conceptually, provision of this service, which extends the U.S. ARPANET to the London TIP, does not affect SATNET operations or performance, except to reduce the number of available time slots in the frame and increase the jitter. However, maintaining this important connectivity while conducting experiments and demonstrations was a delicate process.

The conference session scenario consisted of two main parts. In the first, James Forgie of Lincoln Laboratories conversed in real time with Hugh Gamble of UCL. Voice quality was generally judged quite good, considering the relatively low 2400-bit/s data rate used and the nature of the LPCM equipment. Handover delays were seconds long primarily because of the accumulation of excessively conservative fixed delays at various places in the system to allow for worst-case variance in actual delays. In any deployment of this technology, the handover delay should be much less.

In the second part of the conference session, Hoi Chong of COMSAT linked with Steven Treadwell of UCL via keyboard and video display using the demo terminal and the Clarksburg station. Then connection was made to an ISI host via Clarksburg, SATNET, and Etam to demonstrate access to an ordinary service host. The scenario was designed to illustrate the delays actually encountered as well as some of the remaining problems, mostly congestion. Since simultaneous facsimile transmission and duplication for all members of the audience (numbering well over a hundred) was not possible, the facsimile transmissions were performed early on the morning of the conference session and copied for distribution. The documents transmitted included various greetings, drawings, and timely material from London newspapers. It had been planned to transmit this material during the conference session; however, due to the transmitter adjustments mentioned previously, the UET at Clarksburg could not operate reliably in SATNET during that part of the conference session when the speech demonstration was conducted.

3.2.4 EXPERIENCE GAINED

Besides representing a useful checkpoint in the implementation of SATNET, the demonstration revealed a number of problem areas and issues in the internetworking effort. The main concern

was the severe performance problems with TCP-4 as used with ARPANET hosts, due to packet congestion at points joining links of widely differing bandwidths, such as the SATNET gateways and SIMPs. In the typical scenario, an ARPANET TCP-4 host would emit a pulse of possibly many packets covering the current receive window space. Some of these packets might then be discarded as gateway queues exceeded specified bounds, typically on the order of eight packets. The long transmission delays for individual packets, which then must be funnelled down relatively low-bandwidth circuits, in turn cause the host to retransmit the pulse, often resulting in complete collapse of the circuit and a disconnect for the host. Mechanisms have been built into the gateway system to help avoid these problems, but the current TCP-4 host implementation does not respond to them.

Some congestion problems have been found to originate in the gateway themselves. Although end-to-end FTP traffic, using mostly full packets, experienced few disruptions, interactive TELNET traffic with relatively large numbers of mostly empty packets experienced frequency losses in the gateways, even in simple configurations involving SATNET fake-echo hosts at 4800 bit/s. The cause of these losses has not been determined, but is believed to be related to the very distant host (VDH) or host-SIMP protocol implementation.

The critical impact of the above problems on the end-to-end performance of SATNET is among the most important experiences gained. The problem of TCP-4 performance is endemic in such cases of bandwidth mismatches. Effective and general solutions will be required for TCP-4 to be viable in an internetworking context. However, it is believed that the gateway performance problem involves implementation and should be correctable during development.

3.3 INTERNETWORK FACSIMILE AND MULTIMEDIA MAIL

3.3.1 POSSIBLE CONFIGURATIONS

Presently, two models of Dacom machines are capable of sending and receiving digitally coded facsimile data. One Dacom 500 in the INTELPOST system operates at 50 kbit/s using the CCITT code, while the DACOM 450 used for the recent NTC demonstration operates at speeds ranging from 2,400 to 14,000 bit/s (depending on interface equipment) with a proprietary code. The demo terminal system supports the lower speed machine and provides point-to-point transmission with similar systems. Another DACOM 450 system (at UCL) already has the required characteristics, and three more are being constructed.

The purpose of these systems is to gain experience on the use of facsimile in the internet environment and to develop and evaluate suitable protocols. Scenarios in which facsimile transmission can be important include the following:

a. Electronic mail systems which are primarily intended to augment ordinary mail service with a high-speed document transmission system. Documents are scanned, queued for transmission, and printed at the receiving end. The only input and output are hard copy, and no particular data structure is imposed or implied.

b. An adjunct to existing electronic mail systems such as HERMES or MSG, which exist primarily to record, send, and store messages in ordinary text form between mailboxes attached to the catenet. One application could involve simply incorporating the coded facsimile image directly into the text, perhaps in a coded envelope, so that the receiver could separate the text from the facsimile data and process it accordingly. This is generally

an easy process and can be used for embedded digital speech or other components of multimedia mail.

c. Components in a data base to be used for information archival and retrieval systems. In this case, a well-developed structural component would be necessary to bind the various documents in the data base and provide for their retrieval. Multimedia considerations would not be as crucial, since the binding mechanisms could easily be extended. The catenet would exist primarily to store and retrieve the items in the data base, although its external appearance could resemble case a or b, or a combination of the two.

Of the three scenarios, the second was chosen as most appropriate for study. Valuable experience with scenario a is available from the INTELPOST experiments and demonstrations. The technology to develop an on-line library (case c) would seem to be diversionary with respect to the computer network emphasis of the DARPA project. Further development of case b requires consideration of both transmission techniques and data base management.

3.3.2 SYSTEM ORGANIZATION

After the decision has been made to emphasize multimedia mail applications of facsimile, the questions remain of how to organize the system and how to integrate it with present systems. For example, it would be desirable to interwork with a system such as INTELPOST and to provide for long-term archiving in a large data base. It can be assumed that catenet resources (e.g., ARPANET service hosts) are readily avilable on an internet basis with suitable server programs installed; however, it must be assumed that the facsimile machines are located at arbitrary yet accessible locations.

The most useful configuration for the near term would include a facsimile machine directly connected to a small minicomputer or microcomputer by suitable interface electronics. The computer would then be connected to the catenet. This is the approach chosen in the present effort. However, the facsimile machines can be operated on dial-up or leased lines using suitable modems, and it is expected that remote control of the system and the machines, at least for receiving, is completely feasible. In fact, in normal operation, the machines are often operated unattended at the receive site.

3.3.3 POSSIBLE SCENARIOS

Two possible scenarios are offered, based on the assumption that a message composition, transmission, and storage system similar to HERMES or MSG is available.

In the first, a mechanism exists to store and retrieve facsimile documents at the originating site. Each document has a unique name which can be referenced in an ordinary text message sent to the internet addressee. Upon receipt, the addressee uses ordinary file transport protocols to retrieve the document and print it on the local machine. The document can be queued for transmission at the same time that the text message is sent, as long as distinct connections (in the TCP sense) are used.

The second scenario is the same as the first, except that the document is transmitted along with the text message, perhaps incorporated directly into the text. This procedure is simpler to implement and use, but the transmission time is considerably increased. It is assumed that the text portion of the message is more time-critical and that the facsimile portion can be queued for later transmission.

3.3.4 EXPERIMENT STATUS

Until considerable operating experience accumulates, it would seem prudent to adopt an architecture and implementation strategy which does not preclude either scenario. For instance, the facsimile interface modules should be capable of remote control via a TELNET stream, and appropriate hooks should be provided in message composition and playback modules to permit required manipulations. Prudence dictates the use of as much existing software as possible including TCP-4 and internet FTP and mail servers.

It seems logical to start with either the present UCL (MOS) or demo terminal (DCN) systems on the LSI-11, since they have already been developed to demonstration status. Both systems are built on internet protocols, including TCP-4, and both support the DACOM 450 using the same hardware interface. It is not necessary to decide the type of system at this time, it is important to identify software that can support multimedia operations within the general LSI-11/floppy disk orbit.

During the last year, COMSAT's activity has concentrated on developing foundation software for the demo terminal to support facsimile operations with other catenet hosts. The software under development provides the following functions:

a. Support of the DACOM 450 at the device-driver level for the operating system in use. This generally requires sufficient buffering to cope with latencies elsewhere in the system, unless the interface clock is stopped, which cannot be accomplished with a modem.

b. A mechanism to store and retrieve documents on disk, for example, the ordinary file system supported by the demo terminal. Some enrichment of the file system was necessary to accommodate building queues for deferred transmission.

c. An implementation TCP-4, including an internet layer capable of multiple connections on a per-process basis. The demo terminal already has such an implementation.

d. A simple facsimile server program (PIP) that can be used to control DACOM 450 operation and to supervise data transmission to and from the file system. For unattended playback, this program can be controllable by a TELNET stream.

e. A file-transport server capable of operating within the current internet configuration. This presently consists of "old" ARPANET FTP operated on TCP-4, as controlled by a companion FTP user process.

The current implementation provides point-to-point service only, and does not include provisions for deferred delivery or multimedia integration. Deferred delivery is assumed to be a feature of a planned facsimile user program, which might perform such transmission with a background process. This feature would be desirable for ordinary text mail.

The next step presupposes an operating internet mail system in the LSI-11. This assumes that it is available and provides capabilities to construct and edit messages, forward them using internet protocols, receive and store them, and finally, decode and print/display/speak them. It is further assumed that one of the developmental goals is to provide all of these functions in the LSI-11, without depending on a service host elsewhere in the catenet, except for the internet mail functions.

To effectively manage the limited file storage capabilities in the floppy disk based LSI-11, facsimile files will probably be integrated with ordinary text by storing pointers in the message suitably isolated by appropriate syntax. The facsimile file at the indicated point would be inserted by the mail forwarder as the message is transmitted to TCP. The particular syntax used is thus invisible to the recipient. On the other

hand, the sender could elect to transmit the message and the pointer, and the recipient could request an FTP transfer later. Still another alternative would involve the mailer forking an FTP transfer for the facsimile file when the original message is transmitted. The recipient would then have to collect the facsimile file from its local storage at the time of playback.

These mechanisms have been discussed previously and the problems are well known. This study recommends an approach that provides maximum flexibility and experimentation without excessive implementation difficulties. The only data structure obvious on an end-to-end basis is the syntax of the facsimile file pointer specification as part of an ordinary message. This can be simple and straightforward (e.g., standard file name), and can be changed readily.

A sketch of an implementation plan now underway is as follows:

a. A simple mailer is to be constructed as a user program, which accepts a file containing the message text, adds the header as per appropriate dialog, calls FTP, and initiates the transfer. Initially, it is assumed that the facsimile file specification has already been embedded in the text.

b. It is anticipated that unattended mail retrieval functions cannot be provided when the LSI-11 is being used for other purposes; that is, mail can be received only when the operator specifies. The issue of rendezvousing and the possible use of an intermediate staging host will be deferred. The mail receiver program is a variant of FTP which runs as a user program as specified by the user. It stores received messages as separate files indexed by a master file. This would probably be necessary because of the size of the messages and resulting fragmentation,

as well as to avoid scanning through a single file, as for larger hosts. This also allows message deletion, disk compaction, etc., using standard utilities.

c. A mail reader is constructed again as a user program, which would allow regular selective reading and printing of messages, but would be sensitive to the facsimile-file specification syntax. If a message is detected, the reader signals the local facsimile server to print it, possibly copying it into a separate scratch file so that reading of the original message can continue. Alternatively, if the file were transmitted separately, the mail reader could search for it. If not found locally, it could start an FTP and later request the facsimile server to print.

In summary, the suggested architectural outline does not involve extensive software and can be implemented easily as user-level programs in systems such as the demo terminal. The implementation is straightforward and should be easily modifiable with additional experience and insight.

3.4 DACOM 450 FACS MILE DATA DECODING

3.4.1 INTRODUCTION

As part of COMSAT's effort to support DARPA Internet activities, an algorithm has been developed to decode the data transmitted by the DACOM 450 facsimile machine. This algorithm is intended to allow these data to be represented in other forms for editing, archiving, and transcoding to codes such as the CCITT format. The decoding algorithm is represented as a program written in PDP-11 assembly language suitable for the RT-11 or demo terminal operating systems. This subsection describes the coding technique

used by DACOM, and the decoding algorithm and its performance on typical pages of data. E.A. Poe of DACOM is acknowledged for his assistance in this effort.

3.4.2 DACOM ENCODING ALGORITHM

The encoding algorithm for the DACOM 450 is described by Weber*; however, this article contains a number of errors and omissions, many of which were discovered only after considerable analysis and experimentation. The machine operates over a co-ordinate space of 1,726 by approximately 1,100 pels when in the high-resolution (detail) mode. In the normal (quality) mode, the vertical resolution is halved, so that about 550 lines are transmitted. Finally, in the express mode, about 367 lines are transmitted with missing lines filled in on playback by replicating previous lines.

Data are encoded two rows at a time using a 2-dimensional run-length code. Each row-pair is scanned from left to right and the line-pairs are processed from the top to the bottom of the document. Figure 3-2 shows how the pels are represented. For each i, the vector (x_{1j}, x_{2j}) represents the contents of the jth columns, where x_{ij} can assume values of zero (white) or one (black). Each of the four possible vectors ranging over these values will be termed a state (DACOM refers to them as "modes"), with the succession of transitions between these states determined by the picture content of the particular line-pair. The line-pairs are scanned in sequence with no special end-of-line code in the data itself. For later discussion and comparison with published data, the following conventions will be used:

*Weber, D.R., "An Adaptive Run Length Encoding Algorithm," ICC-75.

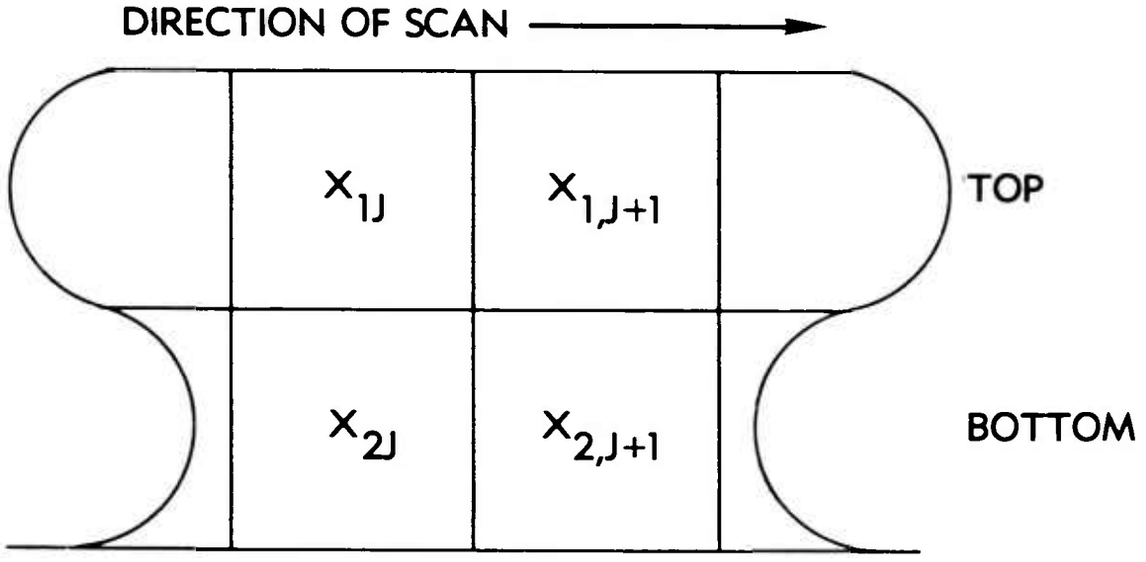


Figure 3-2. Data Representation

Pels (top-bottom)	Vector	State
W-W	(0, 0)	0
B-W	(1, 0)	1
W-B	(0, 1)	2
B-B	(1, 1)	3

The algorithm used by DACOM to generate the transmitted data as the columns are scanned can be described as the nondeterministic finite-state automaton (NFSA) shown in Figure 3-3.

Conceptually, the NFSA starts at the beginning of a page in a designated state, and at a point just after scanning the jth column is in the jth state. It then scans the (j + 1)th column and enters that state while emitting successive bits.

In the states corresponding to W-W (0) and B-B (3), a special run-length encoding technique is used. Two state variables are associated with each of these two states; one variable is used as a run-length counter and the other as the field length (in bits). Upon each entry into either of these two states, the counter is initialized at zero and counts up for every additional column of the same state. At the end of the run, the counter value is transmitted, extending with high-order zeros, if necessary, to fill the specified field length. If, however, the counter exceeds $2^n - 1$, where n is the field length, then a sequence of n ones is emitted and the counter is reinitialized at zero with a field length of n + 1. Thus, if n = 3, a run length of three is transmitted as 010; of seven as 110; and of eight as two words, 111 followed by 0000. The field-length variables are maintained separately for both the W-W and B-B states, and at each reentry to either of these states the previous values are used.

Field-length values are constrained not to exceed seven, so that runs exceeding 127 with n = 7 will be encoded as a separate

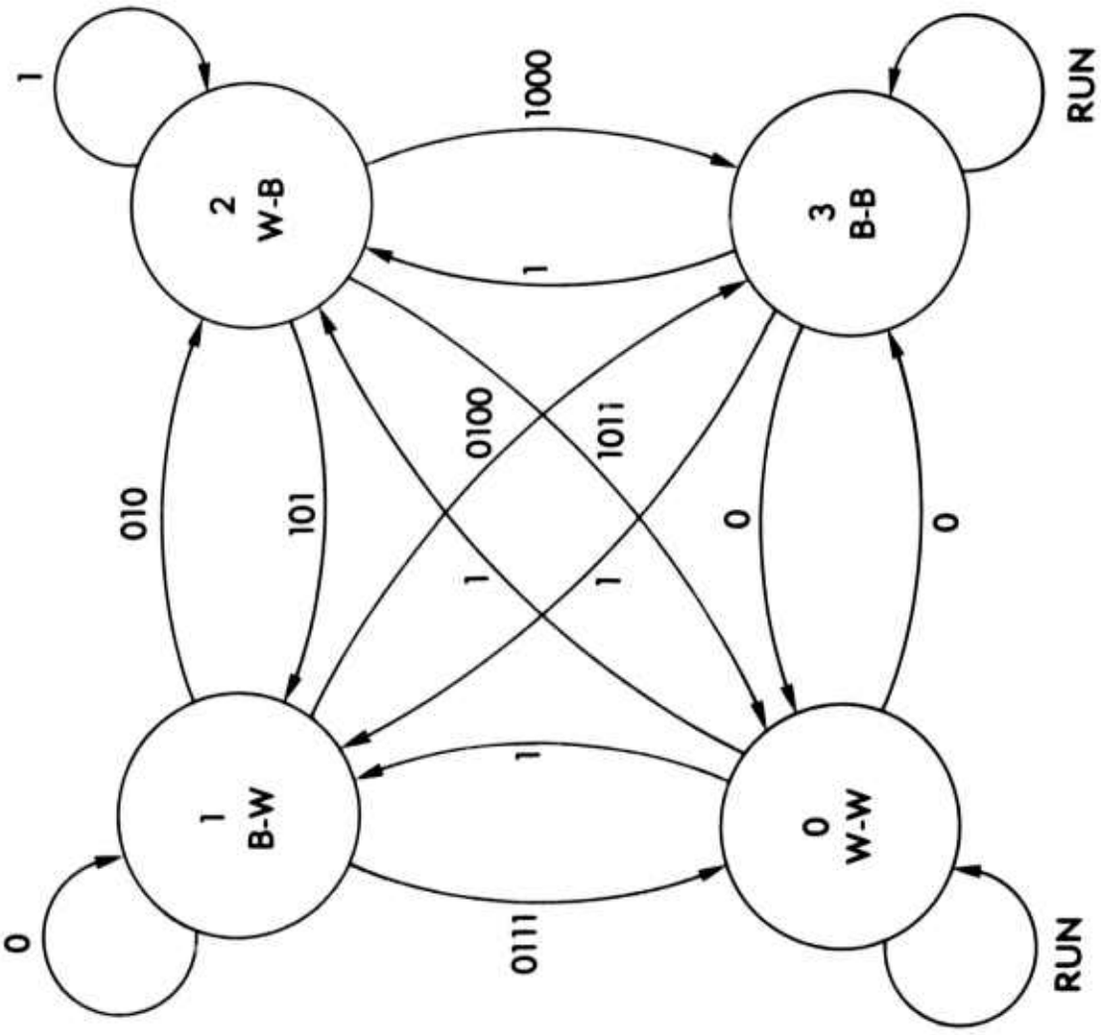


Figure 3-3. DFSA Model of Encoding Algorithm

7-bit word of all ones for each run of 127 except the last, which must always contain at least one zero-bit. The field length n is decreased if the current run has been encoded as a single n-bit field; for n in the range 4-7 when the two high-order bits are zero, and for n = 3 when the single high-order bit is zer. The field length must not be reduced below two bits.

3.4.3 SYNTHESIS OF A DECODING ALGORITHM

The decoding algorithm must be synthesized on the basis of the specified encoding algorithm, since deducing it from the machine circuitry is impractical. However, for speed and simplicity it is desirable that the decoding algorithm be modeled according to the deterministic finite-state automation (DFSA) shown in Figure 3-4. This machine makes one transition for each input bit except for the W-W (0) and B-B (3) states, which must be treated specially. The states are labeled to correspond to those of Figure 3-3, which are numbered from zero to three. The decoded output symbols, which in this case are the columns corresponding to each state, are represented by the states themselves. Upon entry into B-W (1) or W-B (2) a run-length counter is initialized to one. Each traversal of a loop back to the same state increments this counter, and upon exit to any other state, the value of this counter represents the number of columns to be produced. Upon entry into W-W (0) or B-B (3), the run-length counter is initialized to zero and the associated field-length state variable n is established. For each successive n bits of all ones, the counter is increased by $2^n - 1$ and then n is increased by one, but not above seven. If the next n bits are not all ones, the counter is increased by the value represented by the n-bit field plus one. Finally, if upon entry into either state, the next n bits are not

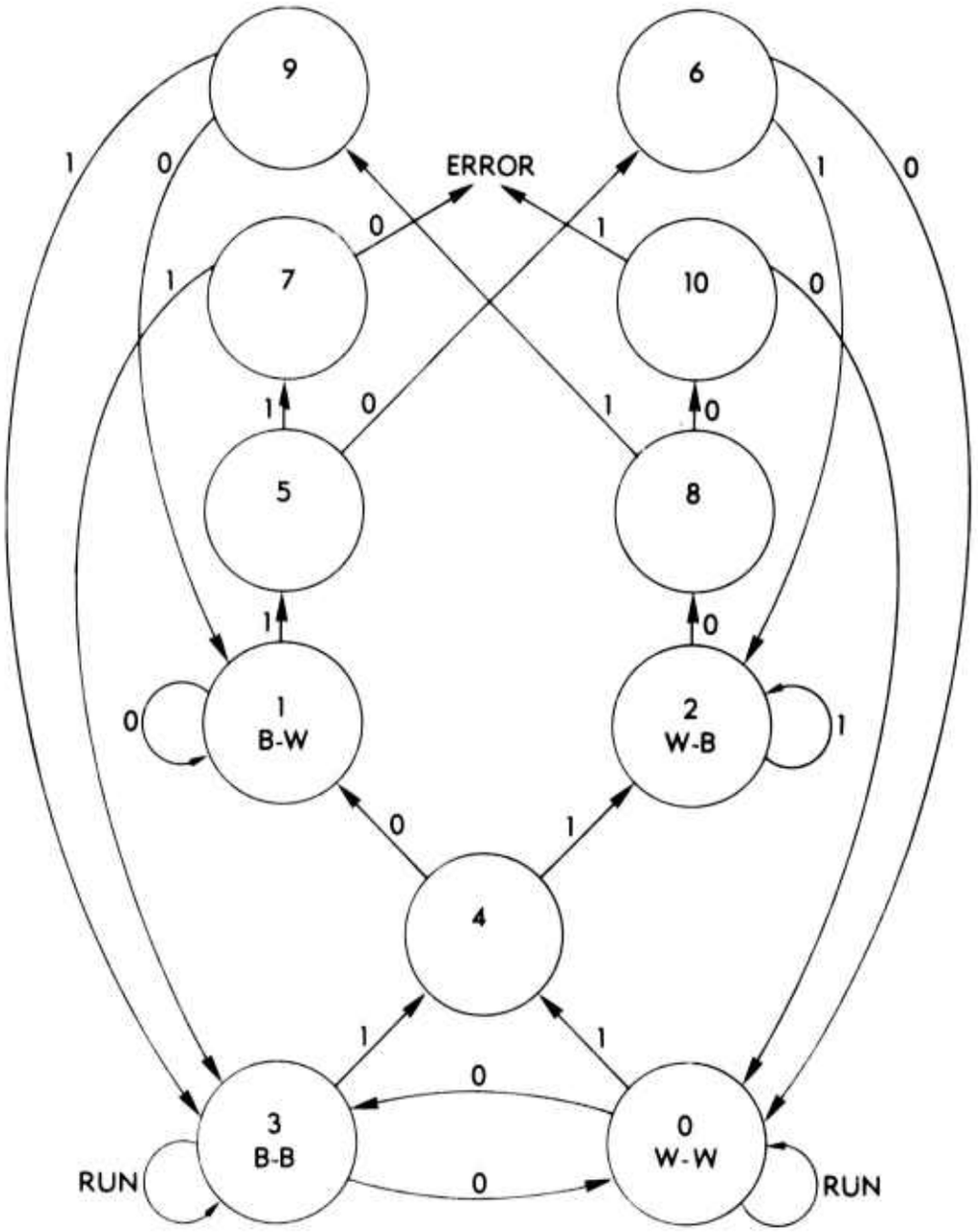


Figure 3-4. State Diagram

all ones; n is decreased by one according to the rule in the preceding subsection.

3.4.4 FORMATTING CONSIDERATIONS

Data are encoded for transmission in 585-bit frames, consisting of a 24-bit synchronization code, a 37-bit header, a 512-bit information area, and a 12-bit checksum. Two kinds of frames are distinguished by the header format: one for setup or initialization, and the other for the data. Serial binary data produced by the encoding algorithm are placed in successive data frames without considering frame boundaries, except that run-length words used in states W-W (0) and B-B (3) are not divided between frames. The header of each data frame, as shown in Figure 3-5, contains the current state values and field-lengths for both black and white runs, along with a count of the bits in the data portion of the frame. It also contains a field specifying the current X position on the page.

These variables, although redundant, help in reestablishing correct decoding if a data frame is lost or mutilated. One of the most delicate and difficult problems encountered while constructing the decoding algorithm was correct synchronization of the header information as the algorithm crossed the frame boundary. To maintain synchronization, the operation of the algorithm must follow exactly that described in the previous subsection.

The algorithm is initialized as the first data frame received after the sequence of setup blocks at the beginning of transmission. The first data frame has a count of zero, indicating that no data bits are in the frame. The second data frame begins the actual document; however, its X position appears to be irrelevant. Instead, it is assumed that the initial X position

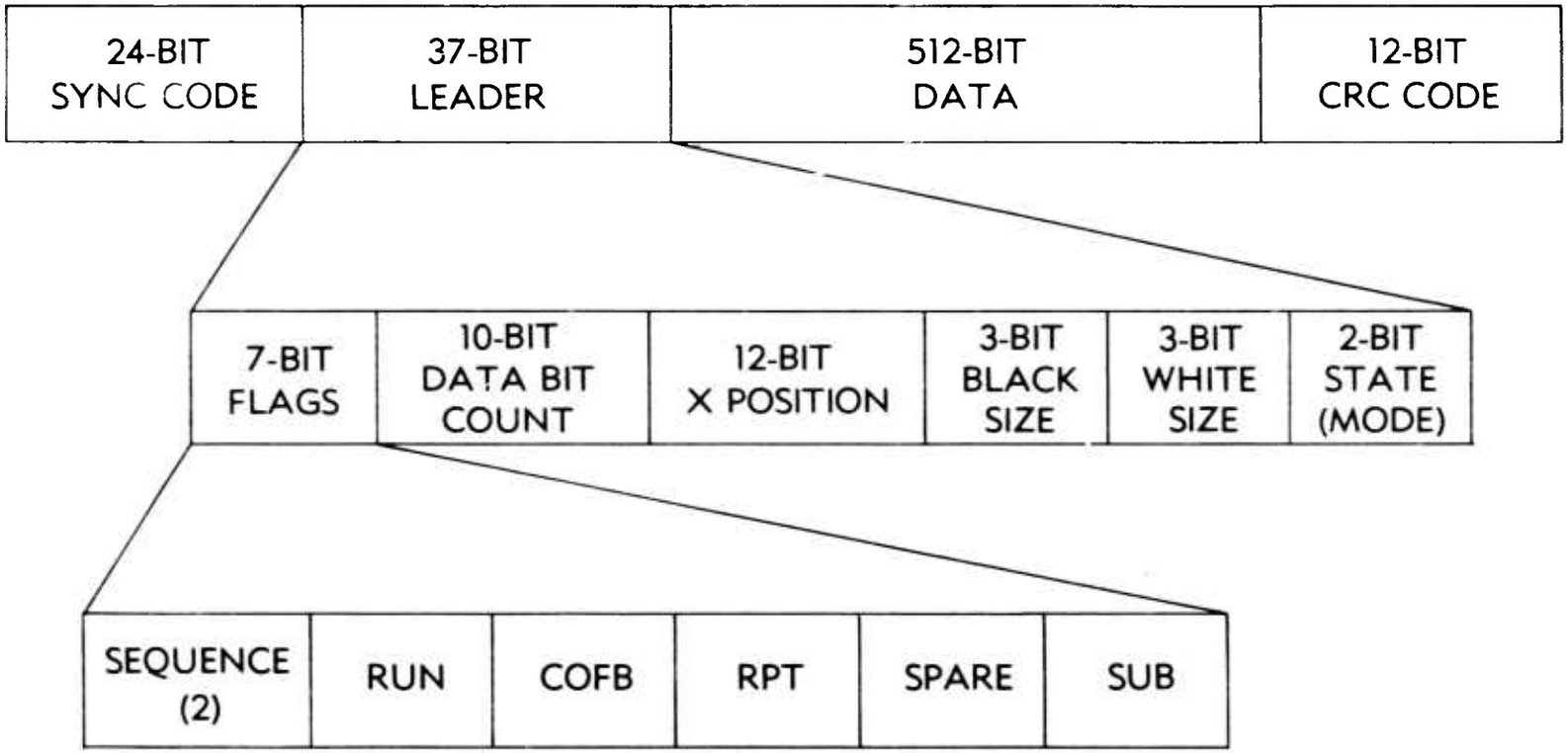


Figure 3-5. Format of Data Frame

is one pel to the left of the right margin (-1 mod 1726). With these assumptions, succeeding X positions of the algorithm and the frame headers agree.

3.4.5 A DECODING PROGRAM

The decoding algorithm described above has been implemented in PDP-11 assembly language for the RT-11 operating system. This program contains extensive features for selectively dumping frames and tracing the operation of the algorithm. It is designed to operate on the raw data stream generated by the machine and does not depend upon any prior reformatting of the data. The input data would normally be in a field which is recorded using the existing DACOM interface software; however, there are no intrinsic factors prohibiting input data from being processed directly from the DACOM machines, except processing-rate limitations.

In operation, the program scans the input data one bit at a time and searches for the synchronization pattern. Note that all data processed are inverted from the natural interface conventions. When a synchronization pattern is found, the header and data positions are extracted and the various state variables are checked and reset, if necessary. Checksum verification is performed according to the polynomial $1 + X^3 + X^5 + X^7 + X^8 + X^{12}$. In the case of setup frames, the format (detail, quality, and express), page length (14, 8-1/2, 5-1/4), and multiple-page indicators are extracted from the data area. Finally, under control of specified options, the header and data portions of the frame are printed with appropriate headings.

The decoding algorithm itself is called for each data frame. It produces an output consisting of a sequence of columns

and run-length pairs which can be used to form bit maps and other representations of the data. Optionally, a printed trace of the operations performed by the algorithm can be produced. With all debugging machinery turned off, the algorithm processes about 4000 bit/s of raw input data on an LSI-11 processor.

A simple bit-map display printout was built into the program to provide a quick-check capability. This display utilizes the 132-column by 102-line space provided by an ordinary line printer and provides for scaling, windowing, and clipping the image in this space. The technique provides for magnifications up to 13 times a window that can be specified anywhere in the page.

3.4.6 DACOM 450 SUPPORT FOR THE DEMO TERMINAL

The DACOM 450 facsimile machine is connected by the demo terminal system with DEC DUV-11 synchronous line interface, and a null modem. In addition, a special software driver for the demo terminal operating system (BOS) has been constructed. The DUV-11 is a buffered, program-controlled, single-line communications interface device and is used to establish a data communications line between a LSI-11 bus and a synchronous modem. The modem eliminator connects the DACOM machine RS-232C port to the DUV-11 interface card. The software drive supports the DACOM machine control procedures, including stripping the duplicate setup blocks and repacking the data into a record compatible with UCL format, as described below. On playback, it unpacks the data from the file and reformats them for transmission, as required. The driver includes features for controlling the timing of the various procedures to provide multiple-rate transmission.

For block transfer to dish or serial devices, the facsimile driver operates with the remainder of the demo terminal system in the same manner as other drivers. Currently, there is no special application program to support facsimile data storage and retrieval other than the standard RT-11 operating system utilities such as PIP (Peripheral Interchange Program). Using PIP, facsimile data are normally transmitted to and from standard system files as formatted by the driver.

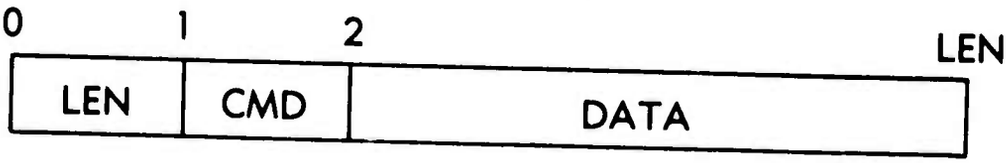
3.4.7 FACSIMILE FILE FORMATS

The DACOM 450 facsimile machine uses a 585-bit block. There are two types of blocks: setup blocks and data blocks, which are distinguished by the SUB flag in the block header. The setup block contains information used by the receiver to establish page length, detail, and similar functions. The data block contains the compressed image data. In operation, the DACOM 450 sends a selected number of setup blocks, then the data blocks, and finally a selected number of setup blocks. The tailing setup blocks differ from the leading blocks only in certain bits which indicate whether the system is operating in single- or multiple-page mode and whether paper is present in the scanner.

The file format used by COMSAT is compatible with that used by UCL and has this form:

-- one leading setup block,
-- all data blocks, and
-- one tailing setup block.

Each 585-bit block has the following format:



Each record is 76 bytes long and has three fields: LEN, CMD, and DATA. LEN and CMD fields occupy one byte each, and the DATA field occupies the remaining 74 bytes of the record. In the LEN field, LEN is always set to 114 (octal), which designates that the length of the record is 76 bytes. The CMD field has the following values:

a. CMD = 70 (octal): the DATA field contains a setup block;
b. CMD = 71 (octal): the DATA field contains a setup block;
c. CMD = 72 (octal): the DATA field contains a tailing setup block, and this is the last record in the file.

The DATA field contains 585 bits followed by 7 bits of zero-padding at the end. The padding bits are inserted to make the DATA field an integral number of 74 bytes.

3.5 CATENET PERFORMANCE MEASUREMENTS

3.5.1 INTRODUCTION

During the development of TCP and internet protocol (IP) modules for the demo terminal and the evaluation of their performance with other hosts on the catenet, there were requirements for a variety of software measurement techniques and capabilities. These requirements were met in two ways: by instrumenting the protocol modules and certain system components directly, and by constructing special measurement programs for end-to-end measurements. This subsection describes the construction of this software and its operation.

3.5.2 SYSTEM INSTRUMENTATION

Facilities for system monitoring and instrumentation have been built into the demo terminal software in many places consistent with the test-bed orientation of the system. These facilities include configuration status, hardware error recording, and resource accounting. In the area of protocol modules, a comprehensive array of packet tallies is recorded and can be displayed at any time. For TCP, a special trace feature provides a continuous record of arriving packets and their interaction with receive-window and sequencing operations.

Figure 3-6 shows an example of packet tallies as recorded by each of the gateway/bridge processes. In general, these processes are responsible for receiving packets, verifying the internet format and checksum, and sending them to the internet process in the local host or relaying them to another host. Three sets of tallies are recorded by the internet process, one by the common processing module and one each by the real-time protocol (RTP) module and the TCP protocol module, as shown in Figure 3-7a through 3-7c. The common processing module is responsible for routing packets to one of the two protocol modules, and for decoding certain gateway-originated packets (source-quench and destination-unreachable) and generating appropriate software signals on the associated RTP or TCP connection. The protocol module tallies reveal the disposition of packets sent to the module, including format and error checking, connection opening and closing, and related events. In the case of TCP, additional information is recorded on packet composition, assembly/disassembly sequencing, and user-interface signals.

For TCP, a special trace feature has been provided, as shown in Figure 3-8. As each packet arrives, the time received, packet identifier, position in receive sequence space, and length

```
Process type: 000027  options: 050400
Last HELLO received: 15:49:24
Congestion controls: 2  60
Input packets:     4312
  leader only:     0
  bad format:      23
  bad checksum:    0
  misrouted:       0                          CLARKSBURG
  HELLO messages:  1031                         LINE
Output packets:    3957
No buffer:         12
Hardware errors:   0
Link resets:       9
Gateway address:   035 000 002 002
Foreign address:   035 000 007 002


Process type: 000027  options: 040000
Last HELLO received: 15:50:07
Congestion controls: 0  30
Input packets:     2000
  leader only:     0
  bad format:      0
  bad checksum:    0                          BACKROOM LINE
  misrouted:       0                             (DIAL-UP)
  HELLO messages:  876
Output packets:    2756
No buffer:         0
Hardware errors:   7
Link resets:       1


Process type: 000027  options: 010400
Last HELLO received: 03:38:15
Congestion controls: 2  60
Input packets:     1495
  leader only:     1267
  bad format:      0
  bad checksum:    0
  misrouted:       24                         NRL LINE
  HELLO messages:  204
Output packets:    1209
No buffer:         0
Hardware errors:   0
Link resets:       6
Gateway address:   012 003 000 050
Foreign address:   012 010 007 010
.
```

Figure 3-6. Gateway/Bridge Statistics

```
Process type: 000026  options: 000001
Input packets:  313
  GGP (3):      1
  GMP (4):      0
  TCP-4 (6):    312
  UCP (7):      0
  XNET-4 (17):  0
  UDP (21):     0
  unclaimed:    1
```

6-2a
INTERNET PROCESS

```
Catenet source quench:   0
Catenet unreachable:     0
Input packets:           1
Output packets:          1
Connections established: 1
Connection errors:       0
```

6-2b
RTP MODULE

```
Catenet source quench:   0
Catenet unreachable:     0
Input packets received:  311
 bad format:             0
 bad checksum:           0
 nonexistant connection: 0
 unacceptable SYNs:      0
 special control/error:  0
Accepted packets:        310
 null (ACK-only):        86
 text stored:            211
 duplicate:              30
 outside window:         7
SYNs processed:          4
 duplicates:             0
FINs processed:          3
Error packets sent:      0
Control packets sent:    9
Text packets sent:       80
Retransmissions sent:    8
ACK-only packets sent:   234
Data avail sigs to user: 210
```

6-2c
TCP MODULE

Figure 3-7. INTERNET Process Statistics

Seq	ID	Start	Length	Window
57151	6144	0	1	488
57318	6656	0	1	486
57532	7680	0	1	488
57868	7936	0	3	442
57911	10752	0	1	481
57926	11008	0	25	459
57949	11264	0	1	463
57995	7936	-74	3	489
58089	11008	-26	25	489
58550	11776	0	1	488
58596	12288	0	1	488
58676	12800	0	1	488
58722	13312	0	3	486
58768	13824	0	1	488
58846	14336	0	1	488
59488	15616	0	2	487
59512	15872	0	3	486
60091	16384	0	2	487
61308	17152	0	2	487
61349	17408	0	37	452
61384	17664	0	21	468
61484	17920	0	118	371
61525	18176	0	39	450
61613	18432	0	118	371
61655	18688	0	46	443
61749	18944	0	108	381
61774	19456	0	2	379
61816	19712	0	37	450
61881	20992	139	39	452
61925	21248	178	46	489
61970	21760	296	46	489
62068	18944	-147	108	489
62124	22272	342	64	489
62167	22528	406	44	489
62190	22784	450	2	489
62231	23040	452	37	489
62265	20224	0	21	468
62298	20224	-21	21	489
62421	20224	-21	21	489
62488	23552	468	21	489
62699	20480	0	97	392
62894	20480	-97	97	489
62936	24064	392	40	489
62988	24320	432	57	489
63021	20736	0	21	383
63145	24576	383	21	489
63206	21504	0	72	85
63256	24832	0	39	118
63305	25344	0	29	135
63353	25600	0	1	198

Figure 3-8. TCP Measurements

are recorded in a circular buffer along with the receive window size after processing. This display has been extremely valuable for detecting network congestion and identifying its causes. An example of this use follows.

Each line in Figure 3-8 corresponds to the arrival of a single packet. The position of the first octet of data, relative to the current receive-sequence number, is shown along with the number of data octets in the packet. Under ideal conditions, with correct sequencing of packets and no losses or duplications, the position of the first data octet will be zero and the number of octets will be subtracted from the receive current window size. (Note that the value shown will be that of the previous line less the number of octets of the current line, increased by the number of octets withdrawn by the user from the reassembly buffer.) Lack of correct sequencing is immediately apparent when the position entry is nonzero. If negative, the packet contains at least some duplicate data; if positive, some data have been lost or received out of order.

From the time-received entry and knowledge of how the remote transmitter packetizes and retransmits TCP data, it is possible to deduce useful information about the characteristics of the network path between the transmitter and the receiver where the trace is made. Additional information primarily for debugging can be extracted from snapshot dumps of control blocks, packets, etc. The multiple-process architecture of the demo terminal system allows this information to be printed locally or transmitted remotely via a TELNET connection as testing proceeds.

3.5.3 MEASUREMENT PROGRAMS

The application programs normally dependent on TCP or IP typically include an intrinsic monitoring and measurement capability for the TCP/IP connections maintained. These programs currently include TELNET, XNET, Server FTP, User FTP, and a special measurement program called PING (Packet Internet Groper). Information recorded by all programs includes connection status, internet addresses, and certain internet options such as timestamps. XNET and PING also contain facilities to measure round-trip transmission times and individual packet counts. Since these programs interface directly with the IP layer and are not dependent on TCP, the intrinsic protocol provides for sequence checking and checksum verification, and the programs themselves record the operational status. All programs provide for connections to arbitrary catenet hosts as well as to well-known or named hosts, including fake-echo hosts in gateways, SIMPs, and other catenet hosts.

PING is designed specifically to measure transmission characteristics over catenet paths containing cooperating fake-echo hosts that are incorporated into the gateways, SIMPs, and other catenet hosts, including those running demo terminal software. Four models describe these fake-echo hosts:

a. Certain hosts, including those of the demo terminal, echo all packets with a specified port (usually port 7), where ports are defined in the packet protocol, such as TCP and UDP.

b. Certain nets, including SATNET, echo all packets to a specified local net address.

c. Certain gateways and hosts, including the SATNET and PRNET gateways and the demo terminal hosts, echo specific gateway-gateway protocol (GGP) packets.

d. Some programs dependent on TCP or IP, including the demo terminal programs listed above, contain echo mechanisms for testing and evaluation.

All demo terminal programs dependent on TCP, including TELNET, Server FTP, and User FTP, can use fake-echo host types a, b, and d. XNET can use b and d while PING can use b, c, and d. In addition, PING can operate using GGP-format packets, but with arbitrary protocol numbers. This feature is now being implemented in cooperation with UCL for SATNET measurements.

The architecture of the existing catenet fake-echo hosts has evolved spontaneously, and without adhering to a particular standard. However, there are two natural mechanizations in which the source and destination internet addresses are simply interchanged, one in which the remainder of the packet is unchanged; and the other in which the internet header is reconstructed according to the current internet option specification. The former mechanization preserves existing options, notably source timestamps, while the latter allows insertion of destination timestamps.

Demo terminal fake-echo hosts of types a, b, and c do not reconstruct internet headers as part of the echo operation; however, type d does. This has been adequate for tests so far, but should be regarded as an incomplete implementation. Specifically, a mechanism is planned with which an existing set of options can be preserved while others are inserted during the echo operation. Thus, destination timestamps can be solicited by a sender while source timestamps are preserved.

The PING program operates on both a user and server basis and can send packets to itself via a remote fake-echo host, either using timestamped packets or end-to-end measured delays. In operation, the program emits a packet, waits for its return, and then transmits another one, so that no more than a single

packet can be outstanding. As each packet arrives, the total round-trip delay is computed and used to update a set of running statistics, which include mean, maximum, and minimum values as well as a histogram showing the number of measurements for each value. These data can be displayed at any time during or after the experiment.

3.5.4 PRELIMINARY RESULTS

A number of baseline measurement runs were performed, both to validate the tools and to gain experience in catenet measurement techniques. The results reported in this subsection should, however, be treated as preliminary, and are expected to be extended and refined as more experience is gained with the complex and sometimes unreliable catenet.

The COMSAT measurement facility, shown in Figure 3-9, consists of various hosts on the COMSAT local network, called the distributed computer network (DCN), connected by up to three paths to the catenet. Each of the DCN hosts, identified as the Playpen (Washington, D.C.), Backroom (Hyattsville, Md.), and Laboratories (Clarksburg, Md.) is a PDP-11 compatible machine with various peripherals, which run the demo terminal operating system (BOS). The Playpen host serves as the hub of the DCN and includes 1822 interfaces to the NRL Pluribus IMP in Washington and the COMSAT Gateway in Clarksburg. The two lines connected to these interfaces utilize error control units (ECUs), which provide a self-contained transparent 1822 connection operating at 4800 bit/s. Connections between the DCN machines currently use dial-up modems operating at 1200 bit/s and a simple protocol using VDH encapsulation, without link-level checksums or retransmissions.

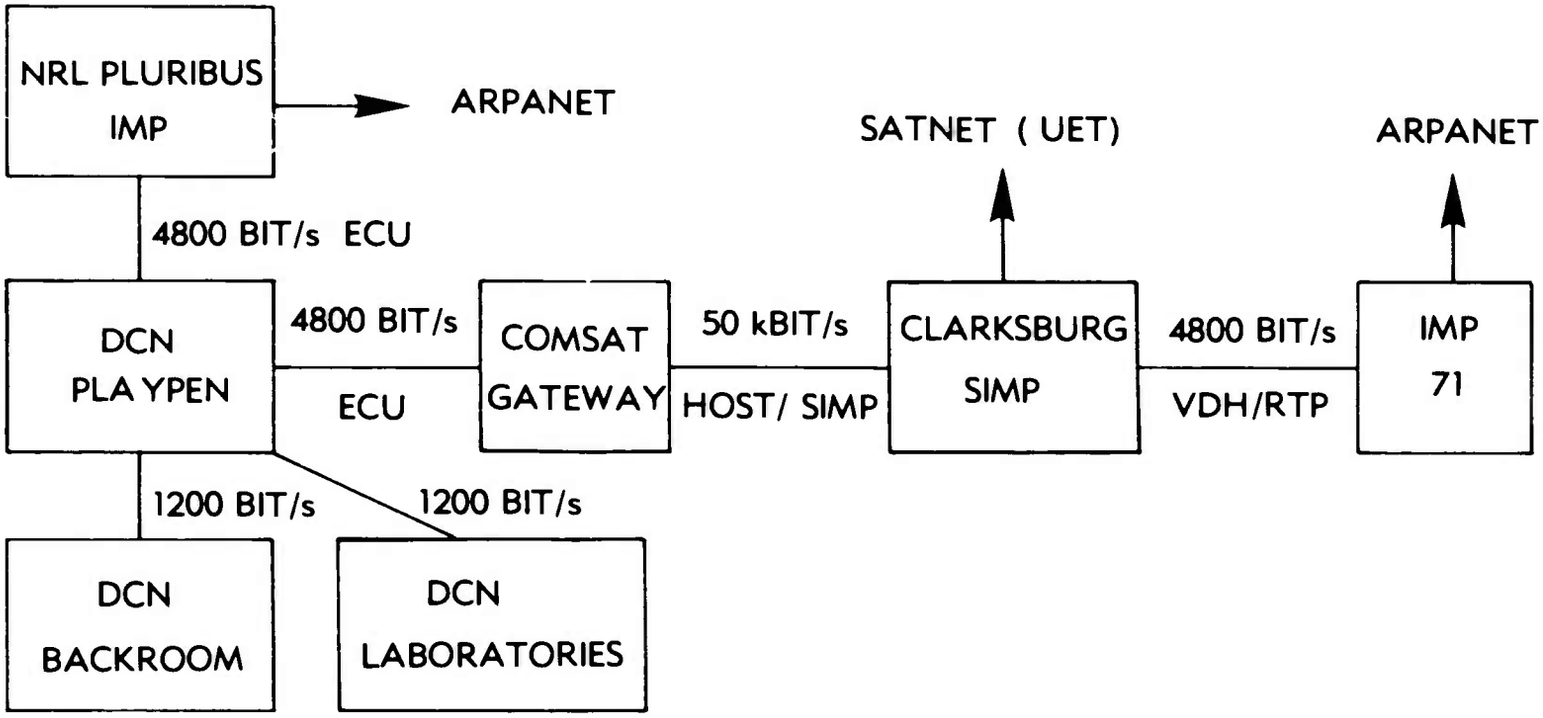


Figure 3-9. COMSAT Measurement Facility

The COMSAT Gateway is connected to the colocated Clarksburg SIMP by standard VDH - Host/SIMP protocol at 50 kbit/s and a null modem. The SIMP can in turn participate in SATNET (but only in mixed-rate mode at 16 kbit/s) or can be connected directly to the ARPANET via a 4800-bit/s VDH/RTP circuit to IMP 71 in Cambridge, Massachusetts. For all experiments described in this report, the SATNET connection was not used and all Clarksburg traffic was routed via the landlines.

Two configurations were used for the preliminary measurements described, one involving the path directly from the DCN to the NRL Pluribus IMP and the other the path via Clarksburg. Although the direct path would seem to be preferred, the path via Clarksburg was used both to calibrate the system for later experiments via the space segment and to investigate the causes of system performance degradation attributed to Gateway-SIMP interactions and the effect of the relatively low bandwidth ECU circuit. In all of the reported measurements, no attempt was made to load the catenet with traffic since only a single packet was allowed to be outstanding at any time. Thus, the measurements can be considered indicative of store-and-forward packet processing, formatting, and transmission delays rather than queueing delays expected under reasonable traffic loads.

Among the many sets of experiments, four were of special interest:

a. A set of runs to calibrate the connections between the DNC hosts and the CATENET, both via NRL and Clarksburg.

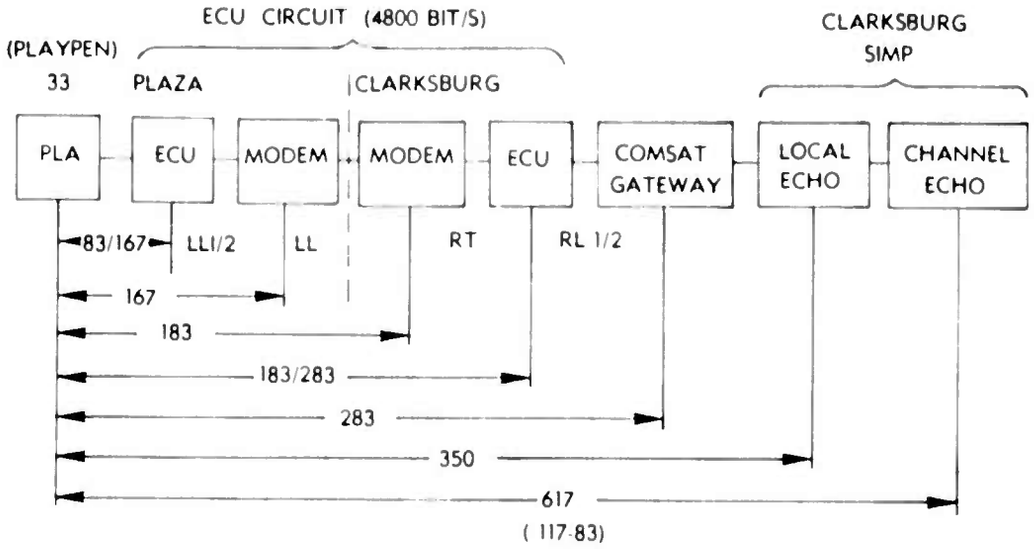
b. A set of runs to determine the CATENET range (round-trip delay) between the Playpen host and the various fake-echo hosts in the SATNET SIMPS.

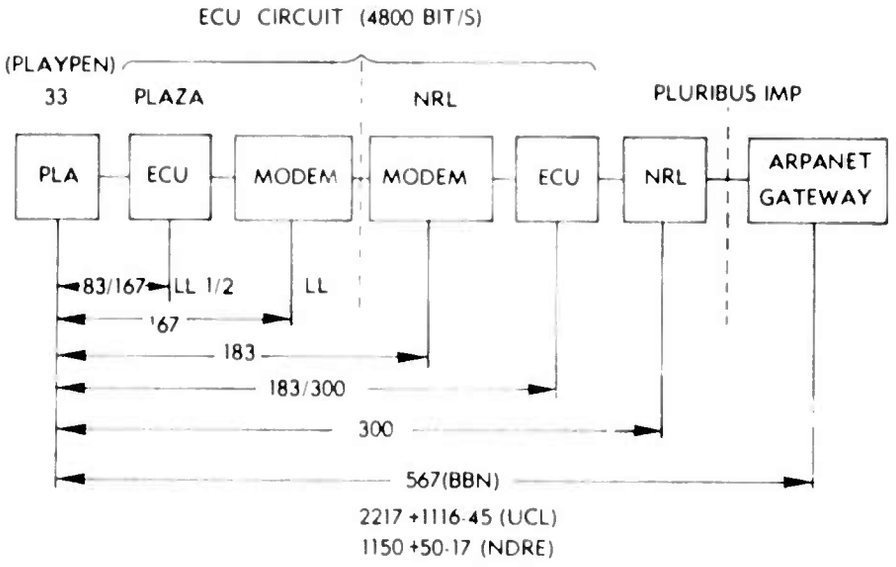
c. A set of runs to determine the range between the Playpen host and selected gateways, including the SATNET gateways.

d. A set of runs to construct delay histograms for the path between the Backroom host and the SATNET gateways.

Figure 3-10 gives the results of the first set of experiments. The top half of the figure relates to the Clarksburg line, including the ECU, COMSAT Gateway, and Clarksburg SIMP; the bottom half relates to the NRL line, including the ECU and NRL Pluribus IMP. The tests were conducted using XNET4 internet packets (52 octets) generated by the Playpen host and looped at the various points shown. In the Clarksburg SIMP, the point labeled "local echo" represents internet (software) loopback, and the point labeled "channel echo" represents external (hardware) loopback via the UET PSP Terminal (not including the satellite channel). The values shown are for an experiment run of 10 packets, with the mean and extrema given in milliseconds.

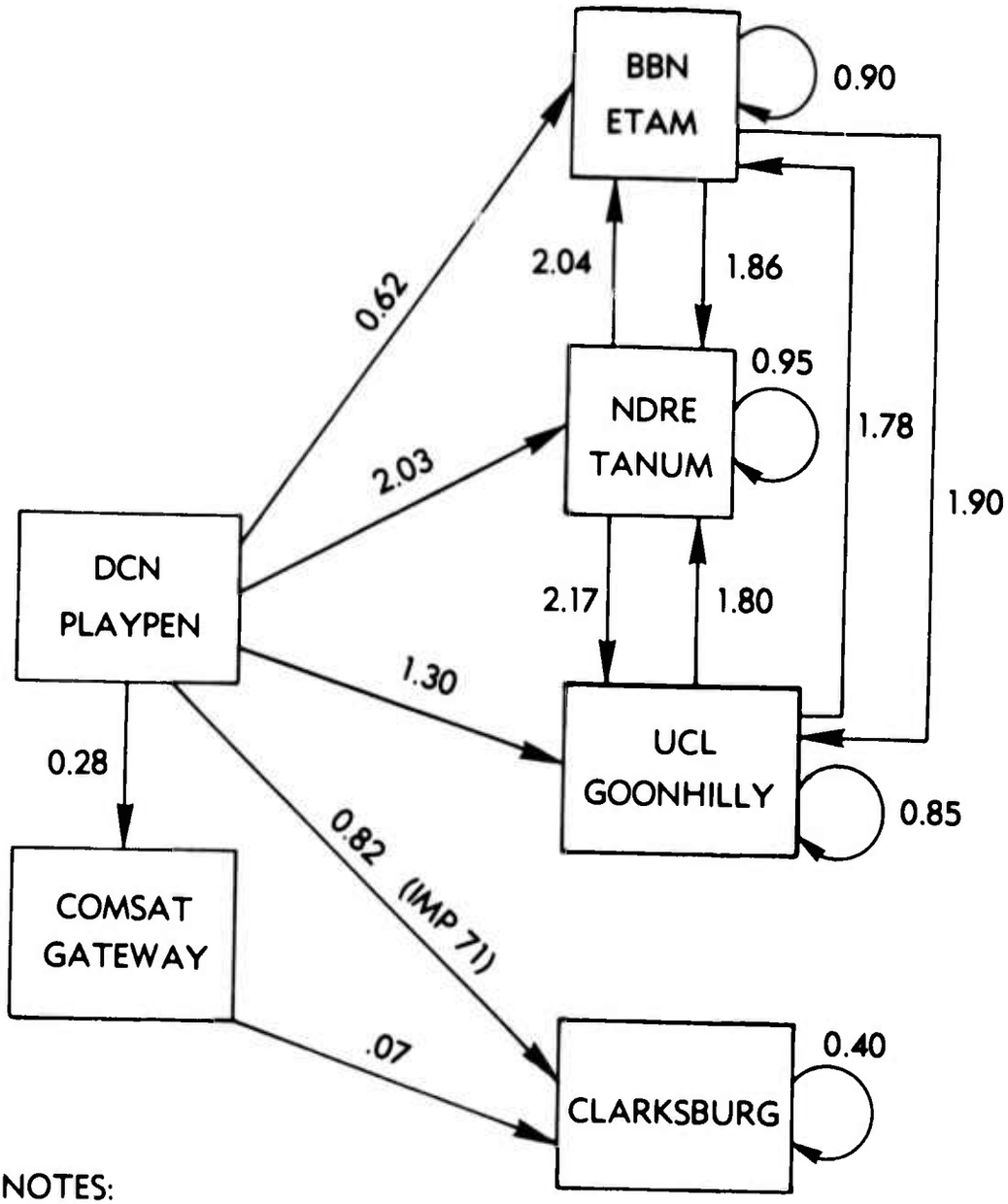
Figure 3-11 relates the results of the second set of experiment runs, which were conducted using XNET4 internet packets (52 octets) generated by the Playpen host and looped at the SIMPs shown. To force the packet routing to follow the paths required, the ARPANET leader was fudged with the selected gateway address (shown in capital letters above the SIMP name) and the internet address in the packet was set to the desired SIMP. Thus, the SIMP-Satellite-SIMP delay between Etam and Goonhilly was determined by forcing the ARPANET path to the BBN or UCL gateway and then setting the internet address as either the Goonhilly or Etam SIMP fake-echo host, respectively. A special fake-echo host was used to measure the delay to and from a SIMP via the satellite. The values shown in seconds are the mean of 10 observations, all of which were made during minimal SATNET usage.





NOTES
1. XNET – 4 PACKETS WITH TIMESTAMPS (52 OCTETS)
2. ALL TIMES ARE IN MILLISECONDS
3. ON ECUS LL-LOCAL LOOPBACK, RC=REMOTE
4. ON MODEMS AL=ANALOG LOOPBACK, RT=REMOTE TEST (DIGITAL LOOPBACK)

Figure 3-10. Local Network Calibration



NOTES:

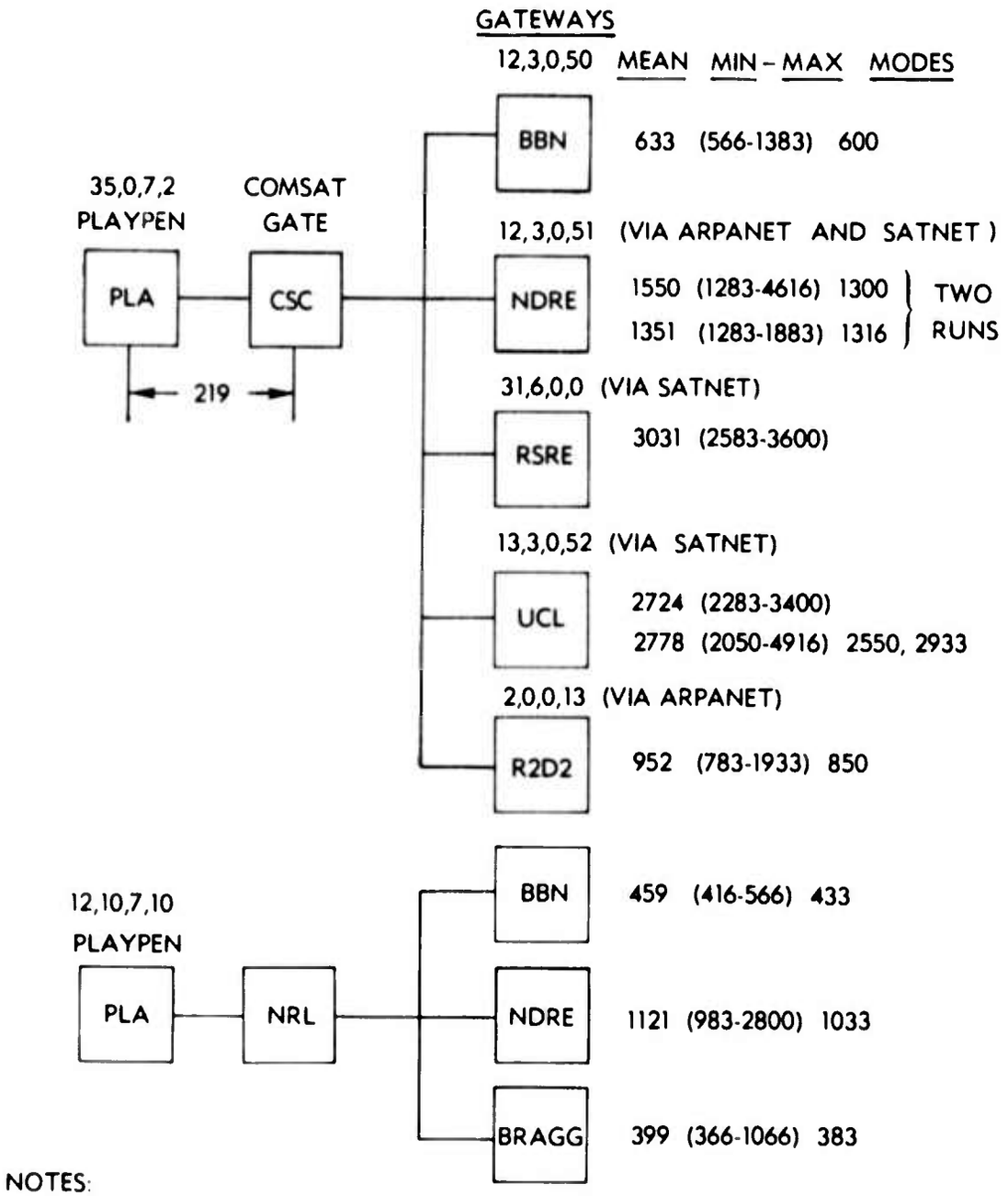
1. XNET PACKETS WITH TIMESTAMPS (52 OCTETS).
2. ALL TIMES ARE IN SECONDS.
3. PACKETS ARE ROUTED TO GATEWAY VIA THE ARPANET LOCAL LEADER, THEN BY INTERNET ADDRESS TO ECHO HOST.

Figure 3-11. SIMP Echo Host Measurements; XNET Packets (52 octets)

Figure 3-12 contains the results of the third set of experiment runs conducted via the Clarksburg path using PING internet packets (36 octets), which have gateway protocol (GGP) format and are designed to be returned by a selected gateway. Each measurement point involved 1000 packets and resulted in the values shown in the following form: mean (minimum-maximum) mode, where mode was determined roughly by inspecting the histogram. In some cases, two modes were apparent, and in others, no mode at all. Near the bottom of the figure are the results of a calibration run using the path via NRL.

The results of the fourth and last experiment runs, shown in Figures 3-13, 3-14, and 3-15, consist of delay histograms for measurements made between the backroom host, the BBN, the NDRE and the UCL Gateways, respectively. The PING program produced these histograms where the delay (milliseconds) is shown along the left margin together with the number of packets observed with that value and a horizontal bar showing the relative number. The label at the top shows the number of samples, in addition to the mean, maximum, and minimum values observed.

The interesting feature of the histograms is their shape. In the case of the BBN gateway (Figure 3-13), which is reached entirely via landlines, the histogram has a single pronounced peak and then tails off as expected. However, in the NDRE Gateway (Figure 3-14), a pronounced bimodal distribution is apparent. This gateway is reached via ARPANET (landline and NORSAR satellite route) paths for the outbound link and returns via SATNET and the BBN Gateway. The explanation of why the bimodal character is observed, where the two modes are about 170 ms apart, is not immediately apparent. A pronounced bimodal distribution is also apparent in the UCL Gateway (Figure 3-15); however, the modes differ by about the value expected for a single satellite hop, 270 ms. This is presumably due to the routing mechanism used by the UCL Gateway,



NOTES:

1. PING PACKETS (36 OCTETS) 1000 SAMPLES.
2. ALL NUMBERS ARE MILLISECONDS.

Figure 3-12. Gateway PING Measurements

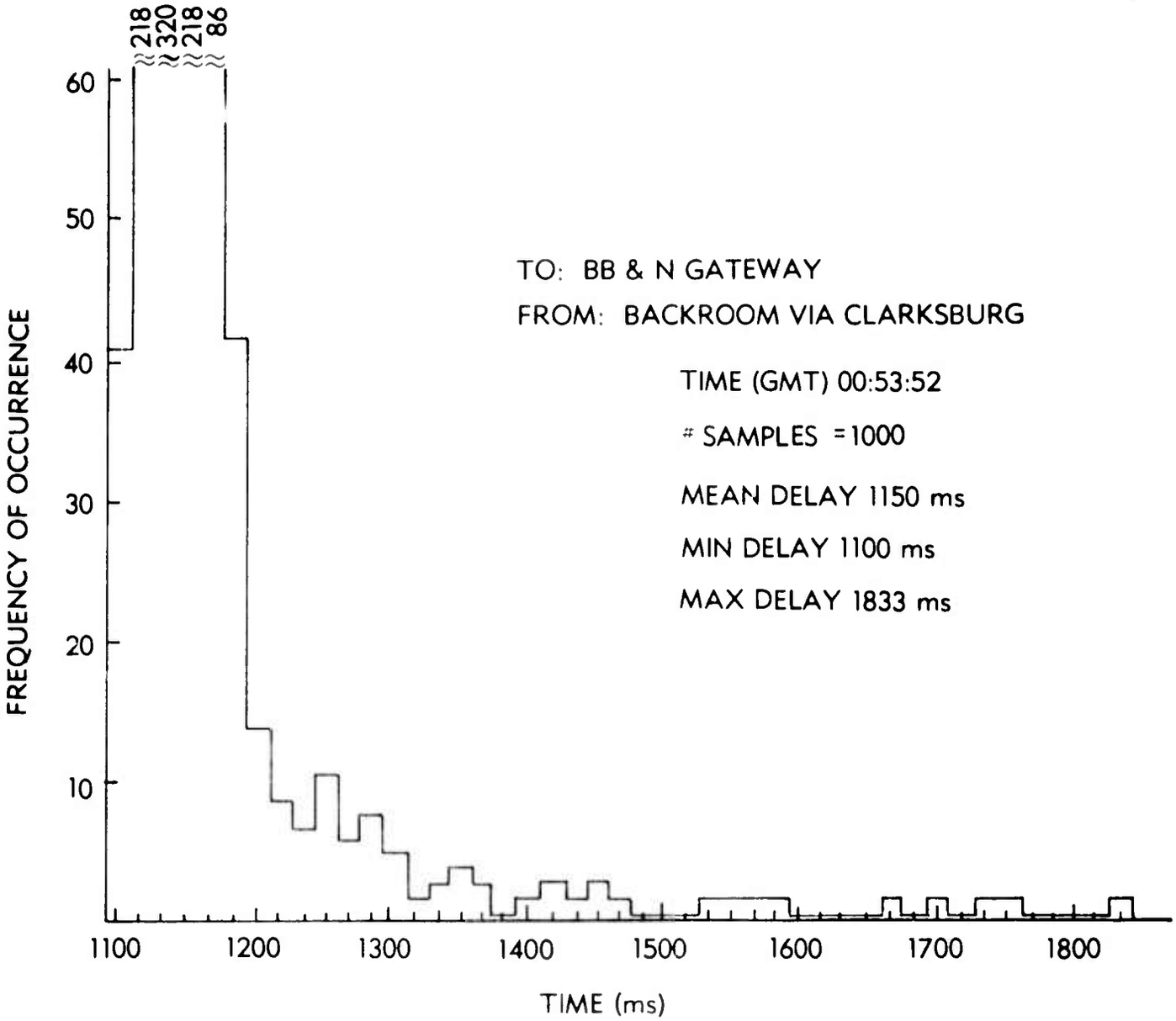


Figure 3-13. Delay Histogram for Measurements Between Backroom Host and BB&N Gateway

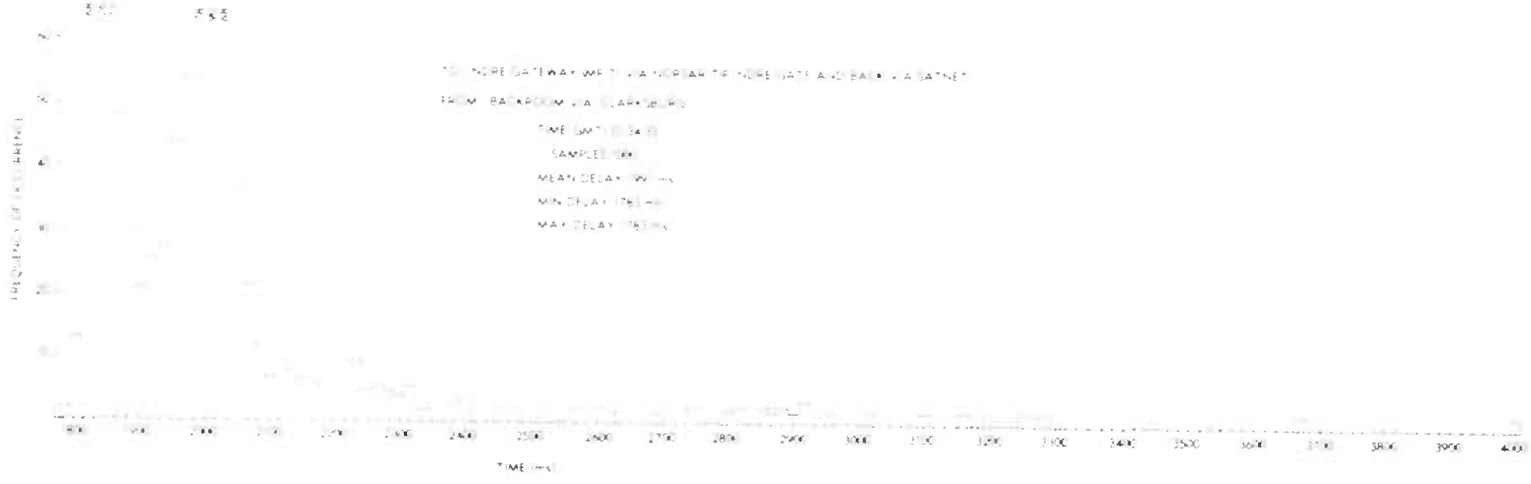


Figure 3-14. Delay Histogram for Measurements Between Backroom Host and NDRE Gateway

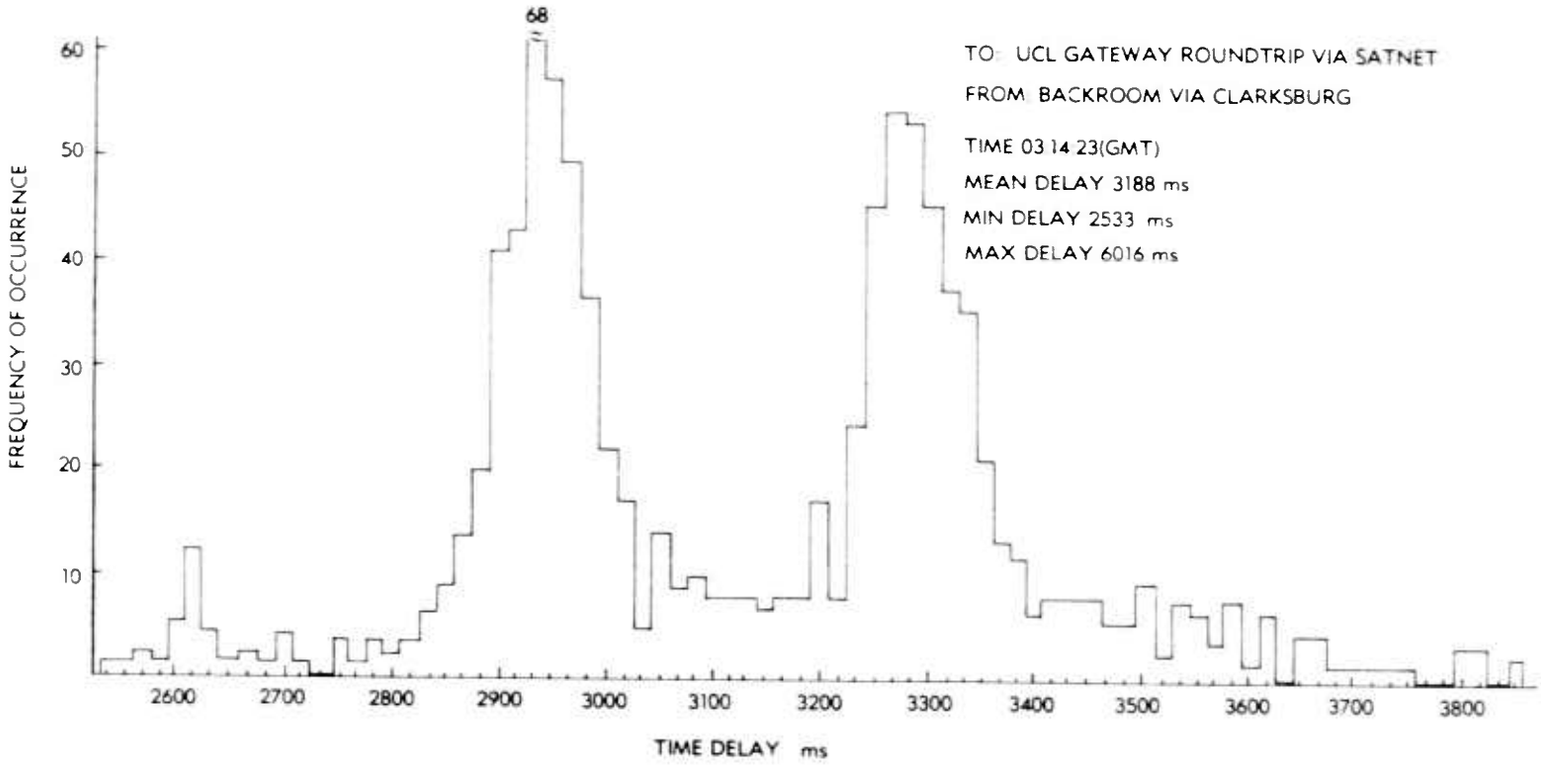


Figure 3-15. Delay Histogram for Measurements Between Backroom Host and UCL Gateway

which, in the case of dual connectivity to the ARPANET (via NDRE and BBN Gateways), routes packets alternately to each gateway in turn. Thus, about half of the packets returned by the UCL Gateway were routed via Etam and the BBN Gateway, which required an additional satellite hop via the NORSAR circuit to the domestic ARPANET.

3.6 INTERNETWORKING AND THE DISTRIBUTED COMPUTER NETWORK

3.6.1 INTRODUCTION

This subsection provides an overview of the issues involved with internetworking using a distributed computer network (DCN) and the basic operating system (BOS) for the PDP-11/LSI-11. The DCN and BOS were originally constructed as part of a research project at the University of Maryland and have been used for a number of years for real-time applications such as intelligent terminals, data acquisition, and machine control. The scope of this subsection is restricted to issues involved in internetworking and interfacing a PDP-11/LSI-11 system that runs BOS (referred to as a hostel), to other networks such as SATNET and ARPANET.

At least five networks have participated in the COMSAT experiments:

a. SATNET: A 64-kbit/s multiple-access multiple-destination network including INTELSAT Standard A earth stations at Etam (USA), Goonhilly (UK), and Tanum (Sweden), and an INTELSAT Standard B station at Clarksburg (UET). SATNET is connected to the DCN via the COMSAT Gateway at Clarksburg, to UCLNET via the UCL Gateway at London and to ARPANET via the BBN Gateway at Cambridge, Massachusetts, and NDRE Gateway at Kjeller, Norway. Also, a path

via the Clarksburg SIMP connects SATNET and ARPANET and another via the Etam SIMP links SATNET and ARPANET.

b. ARPANET: A large-scale packet-switching network including service hosts at BBN and ISI, which represent the principal computing and data base resources available for experiments and demonstrations. The ARPANET is connected to RCCPNET via the BBN-PTIP Gateway at Cambridge, and to SATNET via the BBN Gateway at Cambridge, NDRE Gateway at Kjeller, and UCL Gateway at London.

c. DCN: At the present time, this is the LSI-11 Demo Terminal located at located at L'Enfant Plaza, Washington, D.C. This is designed to support packet speech, facsimile, and interactive computer access using compatible resources on other networks (e.g., service hosts on ARPANET and packet speech and facsimile equipment on UCLNET). The DCN is connected to SATNET via the COMSAT Gateway at Clarksburg and also directly to ARPANET.

d. UCLNET: A local network including a number of special-purpose hosts and gateways to major European public data networks. At least one of the hosts on UCLNET can implement the software developed for the Demo Terminal, and can be used for real-time speech and facsimile traffic in experiments and demonstrations. The UCLNET is connected to SATNET via the UCL Gateway at London and to ARPANET via a port expander and TIP at London.

e. RCCNET: A local network of certain machines at BBN, including the BBN-PTIP. It is used for utility debugging and maintenance of some BBN service hosts and the Clarksburg SIMP and COMSAT gateway.

The software developed for the internetworking community is based on the BOS, which runs on any PDP-11 or LSI-11 system with 28-30K memory, an operator's terminal, a network-interface device, and a direct-access disk. The software has been used in hosts at COMSAT and UCL for SATNET demonstrations. The following paragraphs summarize the facilities available

in the system and explain how they are used to support internet experiments and demonstrations.

DCN facilities include an RT-11 emulator, with which well-behaved standard RT-11 systems and application programs can be executed. A single background job can be executed, together with one or more foreground jobs in a preemptive, priority-driven manner. Standard RT-11 file formats are used, with a special interprocess communications facility similar to that of MOS.

Facilities useful in the internet environment include IP-4, TCP-4, and internet servers for TELNET and a special version of the ARPANET FTP designed to operate with TCP. TELNET, which is a slight modification of the PRNET version allows interworking with appropriate ARPANET host servers and PRNET terminal interface units. TCP-4, also derived from the PRNET version, has been modified to support an internet datagram interface, multiple user processes, and multiple connections per process. It has also been enhanced to provide end-of-letter (EOL) support for use by FTP and to respond to gateway messages.

A special version of the FTP, based on the ARPANET New FTP, has been developed primarily to assist in transferring files among the various DCN hosts; however, it is compatible with the recent FTP specification for TCP and can be used in the future between DCN hosts and others on the ARPANET. The basic philosophy of the ARPANET new FTP design has been followed regarding user/server configuration, command/reply syntax, etc. However, the DCN design allows simultaneous full-duplex file transmission using TCP, where EOL is used as the end-of-file marker, and the data connection need not be closed between file transfers. As in the ARPANET design, the controlling user FTP process, and either or both server FTP processes, can be in different hosts.

In the configuration likely to be useful for internet experiments and demonstrations, a DCN host includes a single background and a single foreground user process, in addition to an

internet process, one or more gateway processes, and various device and system processes, as appropriate for the particular hardware configuration. Approximately 12K words are available for the background process and 4K words for the foreground, with the remaining storage occupied by the procedure code, packet buffers, and various system scratch areas. Each TCP connection requires approximately 1K words in the space of the associated user process, which leaves space for the minimum RT-11 configuration (8K words) and at least two TCP connections in the background process.

In a typical scenario, User TELNET and its TCP connection are run in the foreground process, and the RT-11 TELNET server with its TCP connection in the background. The gateway process presently supports a number of synchronous and asynchronous devices, as well as the SRI-1822 interface using DCN leaders or 96-bit ARPANET leaders. When connected to an ARPANET IMP or gateway, this process functions as an internet gateway; however, the same process can also function as a local network bridge connecting hosts on the same network. A DCN host can include a number of gateway and bridge processes to connect to other hosts and networks; however, only a single gateway process in each DCN local network can be connected to each external network.

The RT-11 TELNET server process can run standard RT-11 programs, including compilers, assemblers, interpreters, and utilities that can run in 8K-12K words. Some RT-11 programs (particularly, BASIC) utilize overlays extensively when run in limited storage, which may diminish the usefulness of some demonstrations. However, it is expected that most experimental application programs such as packet speech, facsimile, and file-transfer packages will not be significantly constrained.

3.6.2 SYSTEM STRUCTURE

To understand the impact upon DCN architecture of internet issues such as addressing and fragmentation, it is necessary to consider the process configuration of a typical hostel. The configuration shown in Figure 3-16 includes a disk, a network interface device, and an operator terminal, each with a device driver process (a set of three processes operating as a unit in the case of the operator terminal). As characters arrive at the operator terminal process, they are accumulated and sent (usually on a line-by-line basis) to a designated user process. Output messages are sent to the operator terminal process as required, with flow control maintained by process-to-process acknowledgments. These messages are in the form of character streams that do not require internal packetizing or explicit system address space manipulations.

Interaction with the file system is aided by a one-segment cache for a volume directory and a set of routines which are part of the RT-11 emulator. These routines communicate requests for block movement of data between main storage and the disk by sending short messages to the disk device driver processes. These processes are required to perform the address translations required if memory management hardware is used.

User TELNET for MOS is run as an application program in a DCN user process. It communicates with the operator terminal using the same facilities as in RT-11 and with TCP using packet-format messages. In many ways, this operation is redundant, since many of the TELNET functions such as line editing and buffering can be handled directly by the console terminal device process, consideration is being given to integrating more of the TELNET functions, such as echo control, into the device process and eventually dispensing with TELNET.

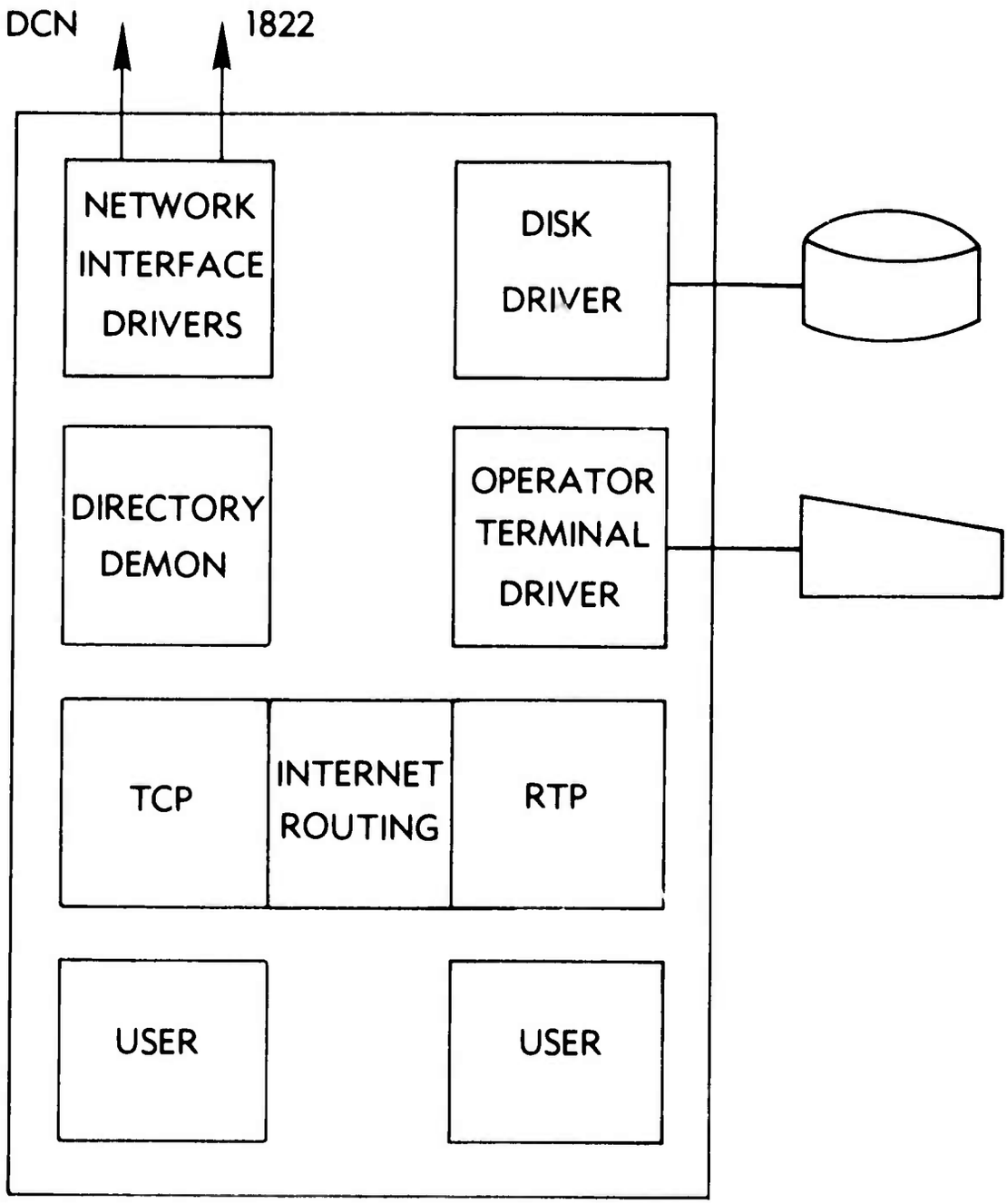


Figure 3-16. DCN/BOS Typical Configuration

The bases for much of the implementation are the TCP-4 and TELNET modules from SRI. These have been modified for use in the BOS which supports the DCN in PDP-11-compatible machines. Generally, modifications required have been minor, preserving the general philosophy and integrity of the original design. The general organization shown in Figure 3-17 includes a single process as the internet attachment for a number of connections. A set of gateway processes interface the internet process with the outside world via device-specific drivers. Versions of these drivers have been constructed for several common communication devices including the SRI-1822 interface.

User processes operate using a set of routines which function as part of the RT-11 emulator package for the BOS. A number of programmed requests (EMTs) are provided to perform the TCP-4 functions of $OPEN, $CLOSE, $SEND, $RECV, AND $INT (the last is not implemented yet).

The internet, user and gateway processes communicate with each other using short messages with functions similar to the $SGNLI primitives of MOS. Where necessary, these refer to packet buffers, which allocated from a common pool, are shared by all processes.

3.6.3 ADDRESSING, SERVICES, AND CONNECTION

The internet address within the DCN is naturally interpreted as the address of an internet process which supports a number of protocol modules, including TCP and RTP. The internet process can possibly be one of several that exist in a DCN. The mapping of the 32-bit internet address to the DCN address is shown in Figure 3-18.

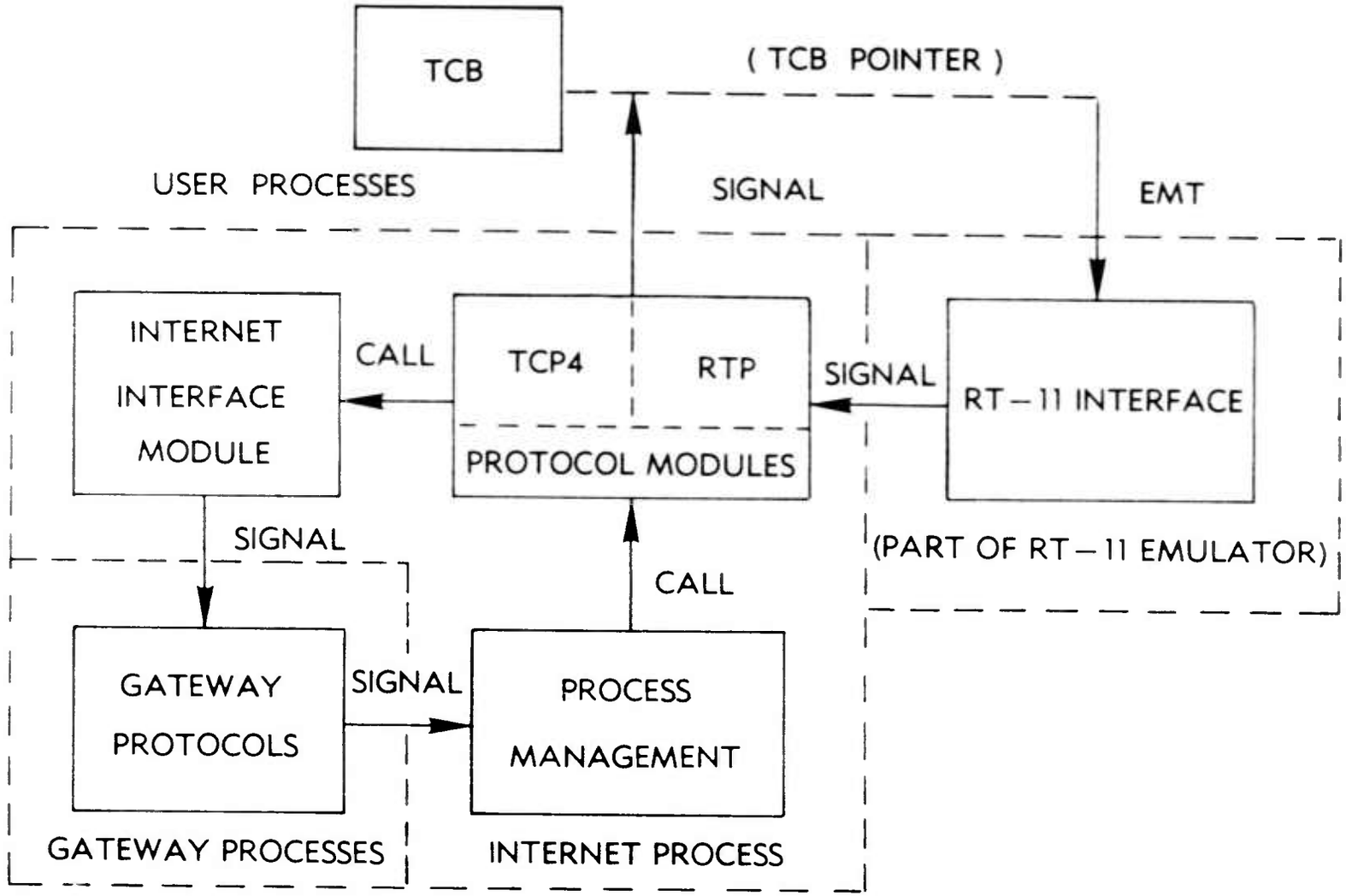


Figure 3-17. System Architecture

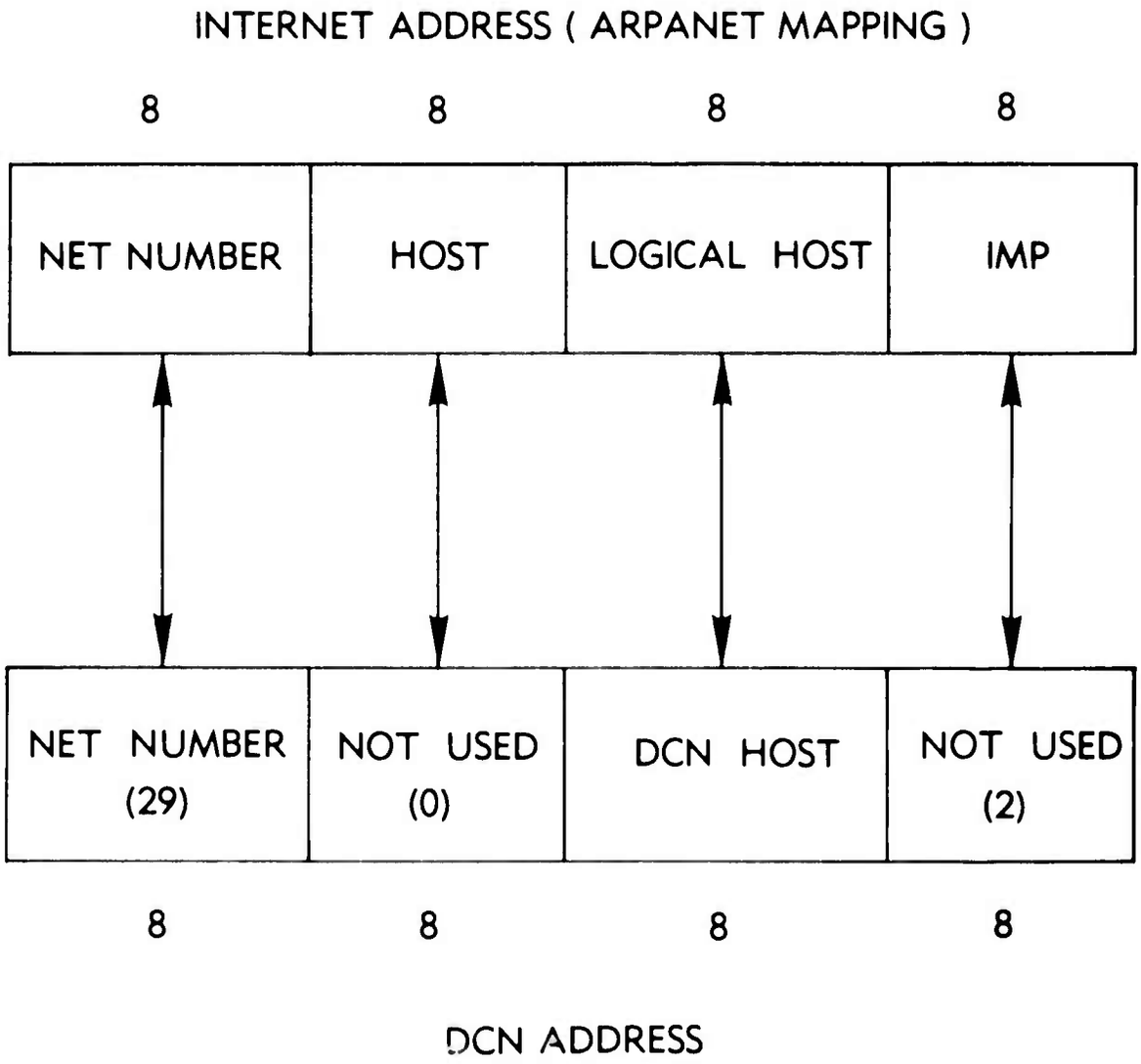


Figure 3-18. DCN Address Mapping

Each internet connection supported by a single internet process is described by a control block called the TCB. Presently two protocol modules (TCP and RTP) can run in such a process. The choice of which module to use for a particular connection is determined by a protocol field in the TCB and by the protocol field in the internet header. User interface to each is identical except that the full complement of connection-related signals is available only with TCP. Data exchanged via TCP are guaranteed reliable and in sequence, while data exchanged via RTP consist of raw internet datagram packets. Present plans call for implementation of certain internet options necessary for SATNET stream access only in RTP, although they may be incorporated into TCP in the future.

The present design can support a single internet process and one or more protocol modules in each DCN host, but it is not necessary for each host to have every module. Data are exchanged using the existing DCN protocols, which assume reliable, sequenced delivery by the same message system used to transport packets and $SGNLI messages. A practical scenario might include an internet process and RTP protocol module in one hostel, a speech module in another, and a control program in a third, with operator terminal and gateway access in any of these or in a separate hostel. In the present implementation, all of these functions in a single 30K-word hostel which provides, for example, about 8K words for a speech application program and a 4K-word for a TELNET user/server.

A particularly crucial issue during design was placement of the TCB, which can range to over 2000 words. In a multiple-connection application such as FTP, these TCBs and packet buffers can consume a large fraction of the available storage. A frequent problem with previous MOS and ELF implementations has been exhaustion of supervisor storage, or storage that is not mapped in the

virtual sense. In the BOS implementation, TCBs are in user space, so that potentially a moderately large number of connections can be sustained without straining supervisor storage, which is then available for packet buffers and other storage structures required to be "wired down" for DMA data transfers. This philosophy requires the internet protocol modules to perform a virtual windowing operation to map the TCB into the address space of the supporting internet process. In the structure shown in Figure 3-17, internet protocol modules are implemented as subroutines using register R5 as a base register. This base can be extended by using a reserved segment in the kernel-space memory-management hardware, if the TCB itself is wired down and not exchanged with secondary storage. Extending the implementation to provide these features would be possible with PDP-11/34 or LSI-11/23 hardware.

3.6.4 SOFTWARE ELEMENTS

3.6.4.1 Introduction

This subsection brifely describes major software elements of the PDP-11 system used at COMSAT for internetworking experiments. Appendices A and B provide additional details.

3.6.4.2 The Basic Operating System

The BOS, which is used in the LSI-11 Demo Terminal, is a multiprogramming executive providing process and storage management, interprocess communications, input/output device control, and application program support. It operates with any PDP-11 or LSI-11 model including at least 8K words of storage,

an operator's console, and a communications device for connection to the DCN. For configurations including a disk, the system supports standard RT-11 operating system compilers, utilities, and user programs in either a stand-alone or network mode. Other features include support for multiple foreground and background jobs which can utilize hardware memory-management features to provide individual virtual address spaces.

Appendix A includes a description of the various BOS components and their operation, and of the various primitive functions and commands used to control network operation and various application programs.

3.6.4.3 Command Language Interpreter

The command language interpreter, as described in Appendix B, interprets and executes typical operator commands to initiate and control system operations. It is structured as a reentrant application program which can be made resident and shared by all user processes or dynamically loaded in the storage partition of each process separately, depending on a choice made at system-link time. The CLI is entered from a user application program via the .EXIT request or as the result of a program interrupt, system malfunction, or control-C interrupt.

3.6.4.4 RT-11 Emulator

The DCN kernel provides a set of features which simulate a standard RT-11 operating system environment. These features, collectively called the RT-11 emulator, allow system and application programs to operate in their native environment, yet

communicate with other systems using the protocol modules described in this report. In general, any RT-11 program that does not directly control a communications or storage device and does not depend on certain infrequently used operating system features of RT-11 can run without change with the RT-11 emulator. This includes all of the RT-11 compilers, utilities, and application programs provided with RT-11 release 3B and 4, together with most programs provided with earlier releases.

3.6.4.5 RT-11 TELNET Server

The RT-11 TELNET server accesses the RT-11 emulator via a TCP-4 connection and a remote TELNET user program, such as the PRNET version which runs in the terminal interface unit. The present server implementation accesses all RT-11 emulator functions, including RT-11 compilers, utilities, and user programs such as FTP, TELNET and network measurement modules.

3.6.4.6 File Transfer Protocol

An internet server has been developed which is designed to facilitate the transfer of files between DCN hosts and hosts on other networks. The server operates using TCP-4 as the transport mechanism and is capable of simultaneous, batched transmission of files in both directions. As in previous versions of the FTP operating on the ARPANET, this server can be controlled directly by a terminal or indirectly by a user program using a simple protocol. In either case, the control connection can be maintained within the local net using DCN messages or remotely using a TCP-4 connection and TELNET.

4. ACTIVITIES UNDER TASK 3: WIDEBAND SYSTEM COORDINATION/INTEGRATION

The wideband, domestic satellite, packet system was studied to determine possible problems. The earth terminal was observed in the Scientific-Atlanta plant, and it was noted that a problem may be encountered in operating the room air conditioner attached to the small "white box" housing the up/down-converter and HPA. The frequent cycling of such an air conditioner tends to promote condensation problems within the equipment in the enclosure for stations that operate in humid climates. It was suggested that the thermostat be set at a high temperature to alleviate this problem.

During review, it was determined that the Linkabit modem requires each of the packets received from the other earth terminals to be within ±500 Hz in RF on a burst-to-burst basis. The long-term frequency drift can be within ±50 kHz.

The requirements for the ±500-Hz frequency tolerance burst-to-burst imply that each of the participating earth stations must transmit with an absolute frequency within ±500 Hz of some nominal absolute frequency. This, in turn, implies accuracy of about one part in 10^7 of the oscillator assembly comprising the transmit chain in each earth terminal. There are two oscillators in the Western Union furnished up-converter and at least one oscillator that determines the modem IF output, all of which make up the transmit frequency determining chain. The frequency tolerance implications on the earth terminal transmit chain have been transmitted to Western Union, which has given assurances that it will be possible to initially set the frequencies to the required absolute values, and that subsequent adjustments due to crystal aging can be made periodically as required.

A wideband system integration meeting was held at DCEC in June 1980, and all participants in the wideband system were represented. Prior to this meeting, it was determined that a number of issues were unresolved: principally, hardware items, cables, and local terminal installation problems. A checklist was prepared listing some 100 items for consideration. During the meeting on June 19, 1980, these items were resolved. A list of 18 action items was prepared and distributed to participants.

5. COMMAND AND MONITORING (C&M) MODULE OF THE PSP TERMINAL

5.1 INTRODUCTION

The concept for a command and monitoring (C&M) module was an outgrowth of the PSP development program. As SATNET operational experience accumulated and the complexity of the new PSP terminals increased, it became evident that the automatic or semiautomatic collection of internal test and performance data would be useful for evaluating system performance and diagnosing equipment faults. Discussions between ARPA, BB&N, Linkabit, and COMSAT concerning possible test and monitoring parameters concluded that provisions should be included for as much system redundancy as possible and for semiautomatic switchover (under SIMP control) to the backup system when practical.

Since system development was dynamic (the concept was being revised during testing and data collection), a hybrid evolved which incorporates both the early (Phase I), and advanced concepts (Phase II) into the final operational configuration of the PSP terminal. Because of this hybrid configuration, test and monitoring data and command functions were included which, in retrospect, were unnecessary or provided little real information. Conversely, test and monitoring (T&M) points, certain command functions, and redundancy, which now appear desirable, were not provided.

The PSP terminal, in its final configuration, consists of both Phase I and Phase II systems at Etam, Tanum, and Goonhilly. A modified Phase II system is included in the prototype PSP terminal which is used with the UET at Clarksburg. The Phase II system is the primary system. The Phase I system, which provides only uncoded QPSK capability, is available as backup in case of a failure in the Phase II interfaces.

In the operational concept, a question remained concerning the processing of T&M parameters and command functions. Presently, it appears that certain C&M functions are primarily of interest to the Network Control Center, whereas other functions are useful only to the earth station personnel or can only be initiated at the earth stations. As an example, earth station operating procedures require that all inactive modems, including the redundant modem in the PSP terminal, be tuned to the earth station test channel (channel 283) and that the transmit frequency synthesizer be turned off. Therefore, the Network Control Center cannot switch modems without requesting that the earth station personnel set the spare modem to the correct channel and activate the transmit frequency synthesizer. As another example, the capability of the C&M module (upon command from the SIMP) to switch SATNET operating frequencies is in direct violation of earth station operating procedures. Thus, this function was not completely implemented in the C&M software; any frequency changes will require manual intervention by earth station personnel.

In the operational SATNET system, remote C&M commands and the retrieval of T&M data will occur over the SATNET channel itself. This is practical as long as the system is performing well. It is also the lowest cost approach, since a separate C&M communications channel would be expensive and, hopefully, seldom used. However, the concept of using the SATNET system for controlling the C&M requires that the net be operational; if the net is operational, there is not much need for many of the remote control C&M functions such as modem switching, modem cross connecting and Phase II and Phase I switching, except for routine tests. These functions will normally not be used unless the system has a catastrophic failure, and, if this occurs, there is no way for network control to communicate with the C&M. Of course, the C&M will still be useful as an on-site control system for routine system checks and data collection.

5.2 C&M FUNCTIONS

The C&M functions can be subdivided into four primary categories: information collection, packet processing changes, system configuration changes, and system troubleshooting. Although certain functions are interrelated, for the sake of clarity this section will not list the secondary interactions, but will deal only with primary functions. The following sections define the various functions of the C&M module, explain their accomplishments, and identify their principal users.

5.2.1 C&M INFORMATION COLLECTION

The C&M module can command the SSI-ϕ2 interface and the modems to collect and transmit the following information to the Network Control Center. With a change in SIMP software and the addition of a medium-speed I/O terminal at the earth station, the information can also be made available to the earth station operator.

5.2.1.1 The Detection Of Anomalies In Transmitted Packet Headings

When enabled by the C&M, the Phase II transmit interface will send one of the following error messages, indicating that irregularities have occurred in the packet header as received from the SIMP and that the packet containing the inconsistent information has not been transmitted. This is performed in real time on a packet-by-packet basis. The anomalous conditions include the following:

a. NO SSDS detected by the interface in the first 32 bits of the packet.
b. PL (packet length) exceeds the maximum allowed by the system (PL_{max}).
c. MODE = 0, an unallowed mode has been detected by the interface.
d. PL > PL_{max} and MODE = 0 have both been detected.
e. MODE = 1 or 7 but CL (coded length) $\neq$ 0 has been detected.
f. PL > PL_{max} and MODE = 1 or 7 but CL $\neq$ 0 has been detected.
g. MODE = 2 or 3 with PL $\neq$ CL has been detected.
h. PL > PL_{max} and MODE = 2 or 3, PL $\neq$ CL has been detected.
i. MODE = 4, 5, or 6, but PL < CL has been detected.
j. PL > PL_{max} and MODE = 4, 5, or 6 with PL < CL has been detected.
k. Excessive duty cycle has been detected. (This condition indicates a malfunction of the SIMP in that packets are being generated at too high a rate.)

All of the above messages indicate an error condition in the packet to be transmitted or in the SIMP that is supplying the packets. When these anomalies occur, the interface will replace the anomalous packet with a dummy packet supplied by the C&M and will transmit the SSDS sequence followed by the dummy packet. At the end of the dummy packet, the appropriate error message is given to the C&M by the monitor. It is important to note that the format of these dummy packets has never been defined by BB&N but provisions for it have been included in the C&M. Originally, the C&M was supposed to load the dummy packet into the RAM of the

Linkabit transmit interface. At present, this function has not been completely implemented, and no dummy packet is transmitted when anomalies are noted in the packets transferred from SIMP to PSP terminal.

5.2.1.2 The Detection of Anomalies (Errors) in Received Packets

During the receipt of packets from the channel, the receive side of the Linkabit interface will look for inconsistencies in the header information and other errors, and will send one of the following messages to the SIMP if the T&M function has been enabled by the CMM:

a. QPSK SOM and uncoded SSDS detected but CL $\neq$ 0.
b. QPSK SOM detected but no SSDS. This message indicates that the system is in MODE = 7 and that a contention for time has occurred.
c. QPSK SOM and coded SSDS detected, PL > CL but the MODE $\neq$ 5.
d. BPSK SOM and coded SSDS detected but PL < CL.
e. BPSK SOM detected but no SSDS or QPSK SOM. This message indicates a contention for time assuming that the system is in MODE = 7.
f. BSPK SOM and QPSK SOM detected but no SSDS.
g. BPSK SOM and coded SSDS detected; PL > CL; MODE = 4 or 6 but no QPSK SOM detected.
h. QPSK SOM and coded SSDS detected but PL < CL.
i. QPSK SOM and coded SSDS detected; PL = CL but MODE $\neq$ 3.
j. QPSK SOM and uncoded SSDS detected; CL = 0 but MODE $\neq$ 1 or 7.

k. BPSK SOM and coded SSDS detected; PL > CL but MODE $\neq$ 2.

l. BPSK SOM and coded SSDS detected; PL > CL but MODE $\neq$ 4 or 6.

m. PREAMBLE detected but no SOM detected.

n. PL > PL_{max}.

All of the above messages are sent to the SIMP in the "TRAP" position of the fourth T&M word; this word also contains the following information:

a. Computed CRC does not agree with received CRC. (A zero indicates agreement, a "1" indicates no agreement.)

b. One-half the number of symbols between the occurrence of the preamble detect signal and the detection of the SOM sequence.

c. The number of disagreements in the correlation of the incoming data with the BPSK SOM sequence (For QPSK transmission modes, this 4-bit word is meaningless.

5.2.1.3 Interrogation of the PSP for System Parameters

Parameters a through e, listed below, are stored in the C&M registers and are obtained by the SIMP by commanding the C&M to dump the appropriate register. Parameters (f) through (k) are available directly from the modem and are obtained by requesting, via the C&M module, that the Linkabit receive interface append the T&M words to the end of the received packet.

a. Length of pure carrier preamble as set by the system.

b. Alternating bit timing recovery sequence length as set by the system.

c. SOM pattern as set by the system.

d. SOM length as set by the system.

e. AGC voltage level from the active modem (packet AGC only). This reading is available independent of the enabling of the T&M words.

f. MODEM AGC on noise.

g. MODEM AGC on packet.

h. MODEM frequency error at beginning of packet.

i. MODEM frequency error at end of packet.

j. E_b (energy per bit as computed by the modem).

k. N_o (receive-noise density) as computed by the modem.

5.2.2 C&M CONTROLLED CHANGES TO THE PSP TERMINAL PARAMETERS

The C&M, upon command from either Network Control or the on-site operator, can change the following parameters affecting the processing of packets.

a. Change Length of Pure-Carrier Preamble. The length of the pure-carrier preamble can be set to 1, 32, or 48 symbols. This pure-carrier preamble is required only for interoperation with the SPADE modems; it will not be used once the final configuration (all PSP terminals) becomes operational.

b. Change Preamble Length. Either 48, 54, or 96 symbol durations of alternating ones and zeros can be selected for operating from large stations; 96 symbol durations can be selected for small stations (i.e., the UET).

c. Change SOM Pattern. Patterns are restricted to one of four possible pattern pairs that are stored in memory. Only one has been defined and implemented at this time.

d. Change SOM Length. There are four possible SOM lengths; only one is being used at this time.

e. Change SOM Detection Threshold. The present number of bit errors allowed in the SOM matched filter is two. This command will permit future changes in the threshold to allow a maximum to five errors (disagreements) in the SOM correlation.

f. Set SOM Detection Window. This function has never been defined and, therefore, is not included in the present implementation.

g. Enable generation of dummy packets. Command dummy packets to be inserted in place of packets with inconsistencies in their header information.

h. Command T&M Words From Modem. Four T&M words are appended to each received packet in place of the 24-bit CRC sequence. When the T&M words are enabled, the CRC is computed by the interface and a single bit in the fourth T&M word indicates whether it is good or bad. The fourth T&M word is generated by the Linkabit receive interface.

i. C&M System Command Functions. Network Control or the onsite operator can command certain operational changes in the PSP terminal via the C&M/SIMP. These commands and their operational characteristics are as follows:

Interface Disable:	The SSI interface will stop transmitting and receiving until an interface enable command is received.
Interface Enable:	Allows the interface to begin transmission and reception.
C&M Disable:	The interface will ignore commands until given a C&M enable command.
C&M Enable:	Enables the interface C&M cababilities.

System Restart: Controlled by either Network Control (via the SIMP), by the on-site operator, or by reset switch on the front panel of the C&M module. This function resets all of the C&M registers, sets the PSP terminal to default parameters and MODE, and resets all other PSP modules.

System Reset: This is a Network Control command (via the SIMP) which resets a default MODE, sets SOM pattern and length, and resets the modems and interface. The C&M internal registers will retain their previous data upon RESET.

5.2.3 SYSTEM CONFIGURATION

This subsection summarizes the PSP configuration to provide background for the discussion of testing and fault location. The earth station portion of the SATNET system consists, in part, of one SIMP (with provisions for a second) and a PSP terminal containing the following major subassemblies:

a. one set of Phase II (SSI-ϕ2) interfaces, including both COMSAT and Linkabit supplied transmit and receive units,
b. one set of Phase I (SSI-ϕ1) interface units,
c. two modems (each modem has a transmit and receive section); and
d. one data test set (DTS) for routine testing and fault diagnosis.

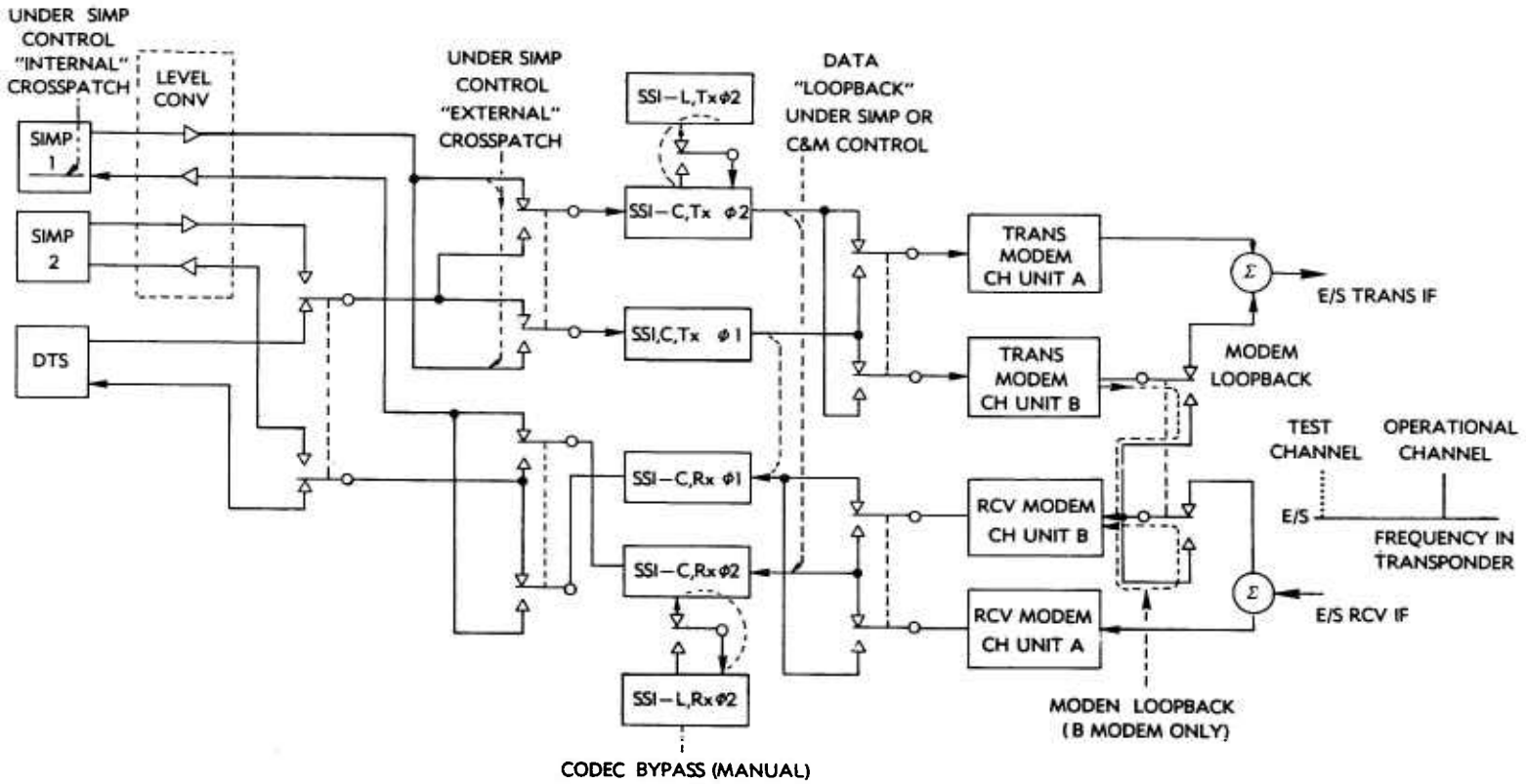
By means of the switch matrix, which can be controlled by the C&M module, the system can be configured into various combinations of these subsystems for operational backup and fault isolation. These options are shown generally in Figure 5-1.

5.2.3.1 Operational Configurations

The final configuration of the PSP terminal is a combination of redundant and backup subsystems. Although it is not fully redundant, there is provision for an independent backup configuration with limited system capability (i.e., the Phase I capability which provides only uncoded QPSK operation).

a. SIMP, SSI-ϕ2, Modem B. This is the standard configuration for the system in which QPSK, BPSK, coded, uncoded, or any combination of the four can be transmitted and received. A backup modem, Modem A, is provided which, on command, can replace Modem B if it is faulty.

b. SIMP, SSI-ϕ1, Modem A (or Modem B). This is the backup configuration which allows continued, but limited, operation if an SSI-ϕ2 interface unit develops a fault. With this configuration, the system is restricted to sending and receiving only QPSK uncoded packets. Since this is more restrictive than the Phase II system, it will be used for the operational channel only when the SSI-ϕ2 is down at a particular earth station. In this configuration, transmissions from other stations cannot be received unless they are Mode 1 (uncoded QPSK); consequently, all other stations in the network must be alerted.



NOTE: FOR SWITCH POSITIONS SHOWN (SWITCH MATRIX MODE 4):

- Φ-2 SYSTEM IS OPERATIONAL (CONNECTED TO THE SIMP) USING THE "B" MODEM FOR BOTH TRANSMIT AND RECEIVE.
- Φ-1 SYSTEM IS CONNECTED TO THE DATA TEST SET (DTS) AND USES "A" MODEM FOR BOTH TRANSMIT AND RECEIVE.

SWITCH MATRIX ALLOWS OTHER COMBINATIONS OF CONNECTIONS BETWEEN SIMP/DTS AND THE Φ-1/Φ-2 INTERFACES, AND THE MODEMS.

Figure 5-1. Configuration of Subsystems in PSP Terminal

c. Cross Connect Modems A and B. The C&M module can change the switch matrix settings to cross connect the A and B Modems, i.e., to allow the use of the transmit side of the A Modem with the receive side of the B Modem, or vice versa. Conceptually, this would seem to be useful since a third level of modem backup is provided. However, it is impractical, since both modems are now engaged and neither one can be serviced. Servicing in the rack requires placement of the defective modem in the channel B position. Consequently, the channel A position must have a good modem before the channel B unit can be serviced. Therefore, it would appear more practical to have a spare modem at each site and to delete the cross connect option from the system.

d. SIMP No. 2 or DTS. The PSP provides a second input port which can be connected either to a backup SIMP or to a DTS. To date, a second SIMP has not been provided; hence the DTS has been permanently connected to the PSP system.

5.2.4 TEST CONFIGURATIONS

In theory, either the Network Control Center or the on-site operator can perform various fault isolation and diagnosis tests via commands to the C&M module. However, since the Network Control Center communicates with the SIMP over the SATNET channel, many of the possible commands could break the link with Network Control. Consequently, most of these configuration changes are practical only for the on-site personnel.

5.2.4.1 IF Loopback Mode

Only Modem B can be switched to connect its transmit output to its receive input. Since transmit and receive are always on different frequencies, this loopback is accomplished by connecting the transmit side through a 45.9875-MHz frequency translator to the receive input. A further restriction is that the modem must be operating in its low band mode.

In this IF loopback configuration, the SIMP/PSP is operating independently of the earth station and the transponder. To the SIMP, the PSP appears to be processing packets normally, except that there is no delay in the round-trip transmission time. This indicates to the SIMP that the PSP terminal is in the IF loopback mode.

In the IF (or modem) loopback mode, the earth station system consists of either SIMP, SSI-ϕ2, Modem B, or SIMP, SSI-ϕ1, Modem B. Successfully processing packets in either of these configurations ensures that these particular subsystems are operating correctly and that any problems are in other parts of the earth station or possibly in the satellite transponder.

In the IF loopback mode, the modem is operating on a noiseless environment, i.e., a high C/N ratio. If the modem becomes inoperable due to excessive noise, then the IF loopback mode, which tests the modem with no noise, would not reveal this condition.

5.2.4.2 Interface Loopback Mode

In the interface loopback mode (also referred to as data loopback, in Figure 5-1), either the SSI-ϕ2 or SSI-ϕ1 can be looped back on itself, thus eliminating the modems from the system. In this configuration, the transmit side of the SSI-ϕ2 or SSI-ϕ1 interface is connected back to its respective receive side.

In the interface loopback configuration with the ϕ-2 system, the SSI-ϕ2 transmit interface is looped back to the SSI-ϕ2 receive interface. The SSI-ϕ2 interface can be operated either with the codec in (a true Phase II system) or with the codec out (a Phase I system using Phase II hardware). Processing packets successfully in either of these configurations demonstrates that the subsystems are operating correctly.

The Phase I hardware cannot be placed in the interface loopback mode by the CM&M. To test the Phase 1 interfaces, the transmit and receive side must be manually connected via a front panel switch operated by the earth station personnel.

5.2.4.3 SIMP Loopback Modes

5.2.4.3.1 SIMP Internal Loopback

In the SIMP loopback mode (controlled by either Network Control or the on-site operator, but not the CM&M), the SIMP transmit port is looped back directly to the SIMP receive port, effectively eliminating the PSP terminal from the system. Since this configuration consists of the SIMP only, successful processing of packets by Network Control ensures that the SIMP is operating correctly and that any problems are outside the SIMP.

5.2.4.3.2 SIMP External Loopback (External Crosspatch)

In the external crosspatch mode (controlled by either network control or the on-site operator, but not the C&M), data are sampled out of the SIMP as in normal operation, and then transmitted back to the SIMP receive port without being processed by

the interfaces. This operation is performed in the switch matrix. A portion of the transmit interface is used only to generate the transmit data. Successful processing of the data in this node ensures that the transmission lines, line receivers and line transmitters, and portions of the switch matrix and transmit interface units are operating normally.

5.2.4.4 Data Test Set Mode

The DTS was originally conceived as an all-purpose, universal test set which could be operated either on-site or by remote control. As the DTS developed, it became evident that the dual option would make it quite expensive; hence, the operational concept was reduced to an on-site test set with only limited remote control capability. In its final configuration, the DTS is very useful for on-site test and analysis. However, the few functions which can be controlled remotely are of limited value in fault analysis, and probably should be deleted from the current system unless the DTS is changed. For example, if the PSP terminal is completely reconfigured to provide full redundancy of the ϕ-2 capability, then the capabilities of the DTS should be expanded to make it a fully remote controlled device. The DTS has been implemented to simulate the SIMP as a source and receiver of packets and to perform limited data analysis.

5.2.4.5 Local Options of the DTS

As an on-site test set, the DTS can perform the following functions:

a. Cause packets to be transmitted with either the F or S options (i.e., short or long preambles, respectively).
b. Vary packet length from 1 to 99 and from 128 to 227 sixteen-bit words.
c. Vary coded length from 1 to 79 sixteen-bit words.
d. Set interpacket interval from 0.1 to 999.9 ms.
e. Select the data pattern in the packet to be either all zeros, all ones, or a pseudorandom sequence.
f. Transmit a single packet.
g. Transmit a preset number of packets up to 9999 and then stop.
h. Transmit a continuous succession of packets (referred to as the free run mode). Some of these "setup" conditions for a test are controllable via the C&M module.

When generating packets in one of the specific formats described above, the DTS will compute and display the following information:

a. the number of packets received with at least one bit error;
b. the total number of bit errors in all packets received;
c. the total number of packets (GOSIGs) generated;
d. the total number of SDS sequences detected;
e. the mode of each received packet;
f. the length of each received packet;
g. the coded length of each received packet;
h. the total number of DLE ETX sequences detected.

At present, none of these DTS test results can be monitored by the C&M module.

5.3 FAULT ANALYSIS CONCEPTS

Fault analysis, isolating the cause of system malfunctions to specific subsystems, can be performed at two locations: the Network Control Center or individual earth stations. Arguments favor both approaches, and since no specific guidance has been received on a preferred approach, provisions for both have been incorporated into the PSP terminal design. The principal arguments include the following factors:

a. Network Control continuously monitors the operation of the entire SATNET system and will probably be the first to detect system degradation or failure at any of the network nodes.

b. Faults at unattended earth stations (UETs) in some future networks can be analyzed only by Network Control.

c. Network Control cannot correct detected faults. Consequently, its fault analysis serves no function other than to switch in backup systems or to notify the affected terminal.

d. Communications between Network Control, the user, and the affected terminal is complicated by administrative procedures and may result in excessive downtime.

e. The monitoring information available to Network Control may depend upon equipment changes made at the earth stations. For example, individual modems must be calibrated to provide useful information on E_b, N_o, E_b/N_o, and the AGC voltage level. When a modem has been replaced at a terminal, Network Control will not be able to effectively use these data unless they have been informed of the change and have incorporated lookup tables for all possible modems in their central files. Therefore, it may be necessary to provide the modem serial number (or other identifier) along with the T&M data.

f. Network control cannot repair or replace faulty subsystems. Consequently, service personnel at manned earth terminals must become involved in any fault correction.

g. Since terminal availability is the principal system performance criterion, fault analysis techniques that minimize downtime will always be preferred.

h. Notwithstanding formal procedures, the user, Network Control, and manned terminal personnel will establish informal communications links in order to minimize downtime and maximize personal convenience.

i. UETs will normally be serviced by personnel of the closest manned facility rather than by Network Control.

j. The DTS, the principal fault diagnosis tool, can be used only by earth station personnel since Network Control cannot receive direct feedback from the DTS test results.

k. Experience has shown that the SATNET system requires more precise power balancing among the participating earth stations than is required by the other users of the SPADE transponder. Earth station personnel have no means for monitoring these levels on other than a once-a-day basis, which is the normal procedure. Consequently, these levels must be monitored continuously by Network Control, and earth station personnel must be informed of necessary adjustments.

5.4 POSSIBLE FAULT ANALYSIS AND ROUTINE MAINTENANCE PROCEDURES

Since the communications channel between Network Control and the C&M module is over the SATNET links, Network Control is limited to performing routine system checks and to analyzing faults which manifest themselves as gradual system degradation. Until the system bit-error rate approaches 10^{-3} to 10^{-4}, Network

Control can communicate with the C&M module, and can attempt to isolate the problem and possibly switch to the backup system. Once a PSP terminal becomes completely inoperative, the Network Control can only notify the earth station that the PSP terminal is out of service.

5.4.1 ROUTINE SYSTEM CHECKS (NETWORK CONTROL)

The following routine system check should be performed weekly by Network Control:

a. With the system operating normally, switch modems to make sure that both are functioning properly. This procedure can be implemented only if the earth station changes its operating procedures to allow the spare modem to be tuned to the operating channel with its transmit frequency synthesizer remaining in the "active" position.

b. Switch the entire system to the backup system (Phase I) to ensure that the backup system is functioning. With the present configuration, this is impractical since the spare modem is not monitoring the channel and has not acquired the correct AGC level. Consequently, at switchover, the entire node will go down while it reacquires system synchronization and the modem adjusts its AGC level. This may require several minutes. Such a long delay could possibly be eliminated by modifying the PSP terminal so that the receive side of the spare modem would be connected to the channel at all times. In this way, the modem AGC level would be correct, and by synchronizing the switchover to the 32-kHz clock, the system should retain synchronization with no significant time delay.

c. Measure modem N_o, E_b, E_b/N_o, AGC on noise, and packet AGC on packets of known length, and check these measurements against the calibration data in the Network Control data bank. Because of possible variations with time (such as hour of the day), these data should be collected at a specific time and logs should be kept to determine the normal diurnal variations to be expected. If any measurement is outside the normal range, a systematic analysis should be initiated to determine the probable cause.

5.4.2 RECOMMENDED CHANGES

The primary problem with system tests by the Network Control Center is the lack of access to the DTS at the earth stations. Network Control can control certain functions of the DTS but it has no direct means of receiving the results, and thus of determining system performance. With minimal functional change, Network Control could exercise each module in the PSP and collect data on its performance. In this manner, weekly checks of the backup system could be performed without direct access to the equipment. As a minimum, the C&M module DTS combination should be changed to incorporate the following:

a. DTS fixed packet mode. Network Control can command the DTS to send a fixed number of packets; however, this number is set by a panel switch and its setting is unknown to Network Control. The DTS should be modified so that when Network Control commands a fixed number of packets, it will automatically switch to a fixed length, format, rate, interpacket interval, and number of packets.

b. DTS receive function in fixed packet mode. The DTS should be modified so that when a fixed packet command is received from Network Control, it will automatically check both the total number of SDS sequences and bit errors detected. The SDS count should then be compared with the preset count; if the SDS count is correct and no bit errors are detected, the DTS should signal SIMP/C&M Network Control (via the SIMP/C&M) that the system is OK.

c. Expanded network tests with modified DTS. By modifying the DTS as described in a and b, the Network Control Center can automatically perform the following system tests currently relegated to on-site personnel:

SSI-ϕ2, Modem B, IF Loopback,
SSI-ϕ2, Interface Loopback,
SSI-ϕ1, Modem B, IF Loopback,
SSI-ϕ2, Interface Loopback with codec off.

d. Hardware Configuration Changes. The system, as presently configured allows the switch matrix to cross connect the modems (Modem A TRANSMIT, Modem B RECEIVE, and vice versa), and provides for a complete interchange (from Modem B to Modem A). However, this switching is accomplished in a manner that restricts the IF loopback to Modem B and always leaves Modem A connected to the earth station IF subsystem. With a minor modification, the system could be changed to switch both the input and output. In this configuration, IF loopback tests could be performed on either modem without physically reversing them, and would allow on-site trouble-shooting without disrupting the operating system. With this change, two additional tests could be added to the above list: SSI-ϕ2, Modem A, IF Loopback, and SSI-ϕ1, Modem (A), IF Loopback.

5.4.3 ROUTINE SYSTEM CHECKS (EARTH STATION PERSONNEL)

A subset of the following routine system checks should be performed perodically by on-site personnel to ensure that the off-line equipment in the PSP rack is performing properly. Switch in the DTS in place of the SIMP and use the DTS in the fixed packet mode to send 999 long packets through the various configurations. Keep a continuous log on each configuration showing the number of packets sent, received, and containing bit errors, and the total number of bit errors and SDS sequences detected. Data should be recorded on the off-line units selected from the following possible configurations:

SSI-ϕ2, Modem B, IF Loopback.
SSI-ϕ2, Modem A, IF Loopback (This requires the physical switching of modems since IF loopback can be performed only in the Modem B position.)
SSI-ϕ2, Interface Loopback with codec on.
SSI-ϕ2, Interface Loopback with codec off.
SSI-ϕ1, Modem A, IF Loopback
SSI-ϕ1, Modem B, IF Loopback.

If any errors are detected during the 999 packet tests, further tests should be conducted to isolate the defective module.

5.5 SYSTEM PERFORMANCE DATA COLLECTION AND ANALYSIS

Although SATNET will become an operational system in mid 1980, there is insufficient data to properly analyze satellite channel performance and to isolate the causes of system degradation. The Network Control Center should begin a long-term data collection

and analysis program to document system performance and to determine which measurements available via the C&M module give the best indication of channel performance. As a starting point, the Network Control Center should collect the data described in the following subsections.

5.5.1 E_b, N_o, AND E_b/N_o

These data are link-dependent and therefore must be sorted and identified by link (i.e., T-T, T-E, T-G, and the six other combinations). They are also nonlinear and depend upon the modem being used at a particular receiving earth station. Consequently, each data point must be compared with the relevant modem lookup table to establish its absolute value.

As a recommended initial step, samples should be taken each second and then averaged into 5-minute groups. The groups should be time-tagged so that these data can be plotted, showing the short term variations as a function of the time of day. The data should also be stored for future retrieval so that longer and shorter periods can be examined, for instance, by the day or month, if necessary.

Since the primary interest will be to correlate the data with channel availability and bit-error rates, it will be necessary to record both missed "hello" packets and packets containing hardware checksum errors for each data point. These packets need only be recorded as good or bad. The missed hello packets and checksum errors should then be summed over the same 5-minute period and the total number of good and bad packets plotted on the same time scale used for plotting E_b/N_o.

5.5.2 AGC ON NOISE

In certain cases, the AGC readings could be quite misleading. For example, the AGC on noise reading will primarily depend upon the time elapsed since a signal was received. The modem is programmed to step up its gain (lower the AGC attenuator voltage) every 0.5 s if a signal is not received. This means that AGC on noise after strong signals could be 10 dB different from AGC on noise when no signal has been present for some time. However, the measurement will give gross information, such as failure of the AGC loop or the LNA; if failures of this magnitude occur, Network Control will not be able to access the data and will have to rely on earth station personnel for troubleshooting. Average AGC on noise will indicate overall system performance and should be monitored during system operation to determine its relation to such performance.

Network Control should monitor AGC on noise during low traffic (receiver gain will be maximum under these conditions) and store the samples for further processing. The initial program should record AGC on noise for each hello packet and then average these readings over 10-minute periods. These data should then be plotted as a function of time and correlated against both missed hello packets (also time-tagged) and other error events as indicated by the T&M trap word (e.g., SSDS, SOM, D-E, and CRC errors). It is also recommended that the 10-minute averages be averaged over 60 minutes and again plotted and correlated against missed hello packets and other error events. This information must be separated by link and identified by modem serial number since the readings of AGC on noise will vary depending on the modem used.

Analysis of these data over several months should show a good correlation between the reading of AGC on noise and system performance, assuming that transmitted power is held constant. unfortunately, transmitted power will vary and contaminate the data. However, it is believed that the information will be a reliable indicator of system performance, and more specifically, of system degradation after a time profile for each link and modem has been established.

5.5.3 PACKET AGC AS A TRANSMITTER POWER MONITOR

If the packet AGC level is monitored and averaged over several packets, and then compared with the calibration table for the specific modem, this reading can give a good indication of the carrier power being received by the modem. If the packet AGC from several transmitting earth stations shows that one transmitter is low or high, Network Control can alert that station to change its power level to correspond to that of the other nodes.

To determine the effectiveness of this approach, Network Control should keep a log of the packet AGC over each link. Readings should be stored only during periods of high traffic density, and a running average over 5 minutes should be computed and stored. These data should then be plotted as a function of time and correlated against missed hello packets which have also been time-tagged and stored. If the analysis of these data shows that power balancing can be effected by Network Control, then either a manual or semiautomatic system should be established to conduct such a routine.

5.6 SEMIAUTOMATIC FAULT ANALYSIS AT THE EARTH STATION

Since the Network Control Center has no method of communicating with a defective SATNET node, it can neither reconfigure the system nor perform automated fault analysis. However, since the C&M module and the switch matrix in the PSP terminal are capable of limited self-diagnosis, it is recommended that a special program be written for the SIMP to cycle the C&M module through specific fault analysis routines under controlled conditions. This program could be activated in three ways:

a. The on-site operator notices that traffic is not flowing through the PSP/SIMP and commands the SIMP to run its diagnostic routine.

b. The SIMP, in monitoring hello packets, will immediately know that the node is not receiving traffic, and after a fixed time delay, could start its diagnostic routine.

c. The SIMP could automatically send periodic "handshake" messages which require an acknowledgment by each node. If no acknowledgments are received, the SIMP could decide that the node was not operational and start its diagnostic routine, which could take the general form shown in Table 5-1. The SIMP could then print its fault analysis and request specific action by the on-site personnel. If no fault is found, the SIMP would then ask the operator to check its analysis or proceed with more complex troubleshooting.

Table 5-1. Automatic Fault Diagnosis Routine for SIMP

a. Fault Condition: SIMP has processed no hello packets from any node for † minutes.

Step	Action Taken	Decision To Be Made
1	Switch Modems A to B or B to A*.	If hello packets are heard within 1 minute, alert operator that modem is defective. If no hello packets are heard, go to step 2.
2	Switch in Modem B and IF loopback. Send dummy packet from SIMP.	If test packet is received by SIMP, alert operator that earth station channel is out. If test packet is not detected, go to step 3.
3	Switch SSI-ϕ2 (codec in) to interface loopback. Send test packet from SIMP.	If test packet is received by SIMP, alert operator that both modems are out. If test packet is not detected, go to step 4.
4	Switch SIMP to SIMP external loopback. Send test packet.	If test packet is received, alert operator that SSI-ϕ2 is out and go to step 5. If test packet is not detected, go to step 6.
5	Configure system to SSI-ϕ1, Modem B.	If hello packets are detected within 1 minute, alert operator that system is in backup mode and that SSI-ϕ2 is out. If no hello packets are detected, go to step 7.
6	Repeat step 4.	If no test is packet detected, alert operator that SIMP is defective; start SIMP diagnostic routine.
7	Configure system to SSI-ϕ1, Modem A.	If hello packet is detected within 1 minute, alert operator that system is in backup mode and that SSI-ϕ2 and Modem B are out. If no hello packet is detected, go to step 8.

*Cross-connection of Modems A and B could be performed next to isolate specific subchannel fault.

†A specific time interval to be determined (for example, 10 minutes).

Table 5-1. Automatic Fault Diagnosis Routine for SIMP (Continued)

a. Fault Condition: SIMP has processed no hello packets from any node for † minutes.

Step	Action Taken	Decision To Be Made
8	Switch in IF loopback. Send test packet from SIMP.	If test packet is detected, alert operator that SSI-ϕ2 and earth station channel are out. If test packet is not detected, go to step 9.
9	Switch in SIMP, SSI-ϕ2 (codec out), interface loop-back. Send dummy packet from SIMP.	If test is packet detected, alert operator that SSI-ϕ2 (codec out) and SIMP are OK. Remainder of system is down. If no test packet is detected, switch to "interface disable" and alert operator that system is off the air. Go to step 10.
10	Switch to standard configuration (SSI-ϕ2, Modem B); request AGC on-noise reading from Modem B.	Average AGC on-noise readings for 1 minute (sixty samples). Check reading against "normal" for specific modem. Notify operator of results. Go to step 11.
11	Switch to Modem A, request AGC on-noise reading from Modem A.	Same as in step 10.

†A specific time interval to be determined (for example, 10 minutes).

5.7 RECOMMENDED PROCESSING OF T&M DATA

This subsection describes the recommended initial processing of T&M data. The goal of this processing is to provide information relative to SATNET link conditions on a regular basis:

a. average E_b/N_o for received packets by link at each received earth station;
b. the cumulative distribution of E_b/N_o readings of received packets by link; and
c. the average frequency offset of received packets, sorted by link.

It is recommended that the same averaging period be used for all of the above measurements (10 minutes seems to be a reasonable initial value).

Each packet received by the jth receiving earth station is assumed to contain the following information:

$$Pk\{i, PL, t \underbrace{AG_1, AG_2, F_1, F_2, E_b, N_0, W_4}_{\text{T\&M words}}\}$$

where

i = identifier that indicates the transmitting station
PL = packet length
t = "type" of packet (i.e., hello, control, or data).

Also, the SIMP at the receiving station knows "j" (the receiving station identifier) and also can measure elapsed time, T. The SIMP also knows which modem is on line at the receiving station

so that it recognizes the modem identifier "MOD" (this could be the serial number of the 7 Linkabit modems, two per large station and one at the UET).

The processing shown in Figure 5-2 is performed for each link; for example, at Etam it would include packets from Etam, Goonhilly, and Tanum. The processing deals only with one of the frequency offset bytes, F_2 (final frequency offset), and the E_b and N_o bytes. The frequency offset byte is recursively filtered directly to give OUT1. The time constant 64 may have to be selected by experiment, but the 1/64 multiplier seems to be a good starting value. The scaled ratio of E_b/N_o, RAT, is formed and corrected according to packet length (PL) and modem serial number (MOD). This correction factor would be provided by the Table ADJ(PL,MODEM). Corrections for packet lengths of 8, 16, 32, 64, 128 words should give sufficient accuracy. After correction, the ratio RAT is recursively filtered using the same averaging time.

The 8-bit value of RAT is used to increase the histogram value for that particular index by one unit. To allow the histogram to accumulate continuously, a normalization is shown that ensures that the "bin" with the most members never overflows. (It may be possible to use 8-bit words for the histogram populations.)

After updating the averages of RAT and F_2 and updating the histogram, the processing returns to wait for the next packet. When the reporting period has elapsed (10 minutes is suggested), the following additional processing is performed:

a. Average F_2 is converted to Hz and outputted by the link.

b. Average RAT (i.e., average E_b/N_o) is converted to dB and outputted by the link.

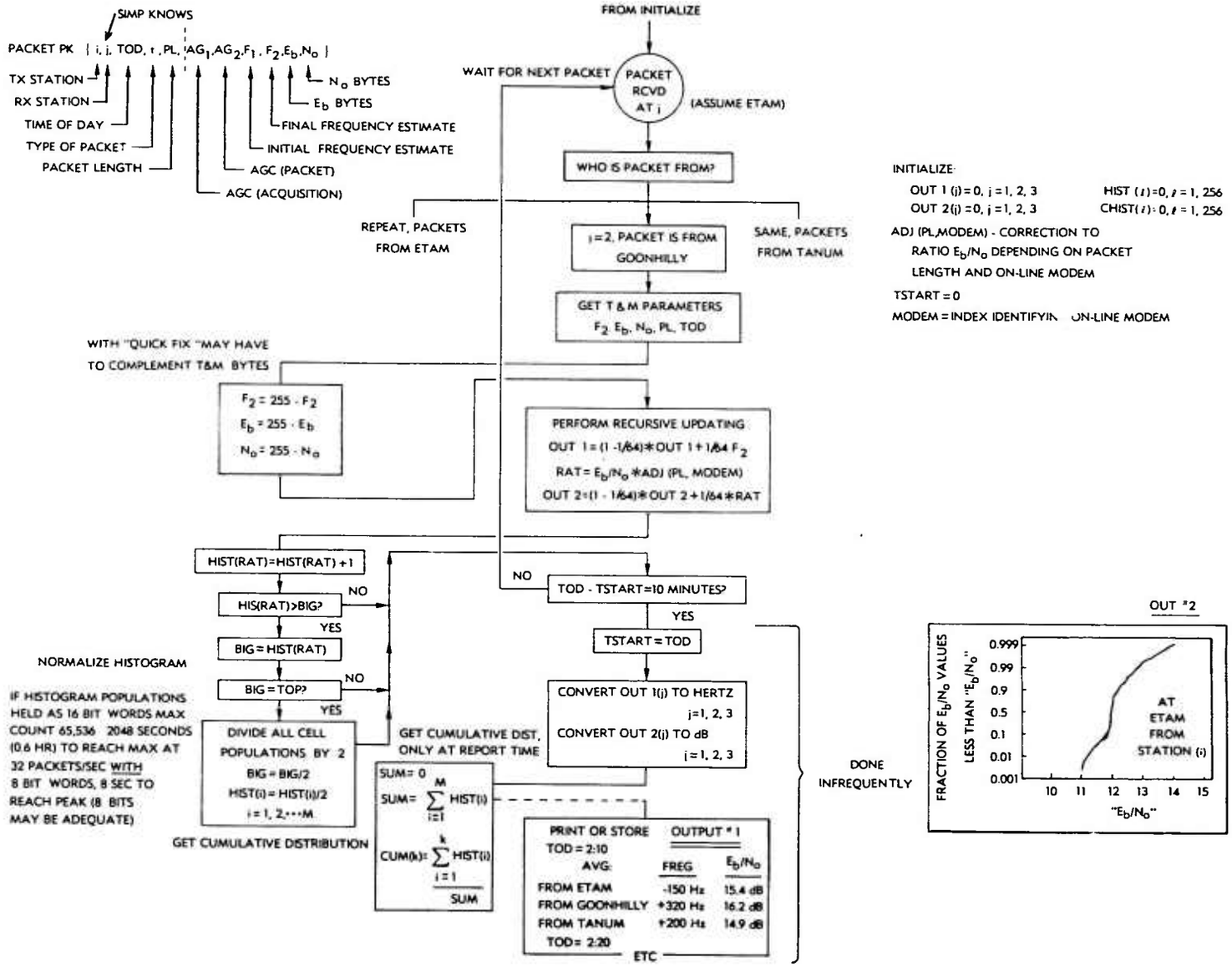


Figure 5-2. Recommended T&M Processing

c. The cumulative distributing of E_b/N_o is formed and a table (or plot) of "fraction exceeded" is provided versus E_b/N_o in dB. These cumulative distributions are provided by the link. The reporting period for these values may be longer than 10 minutes (hourly may be adequate).

It is tacitly assumed that this processing would be performed by each SIMP. If this is not possible, an 8-bit microprocessor could probably accomplish the job (with some compromises). It is even possible that the Motorola 6800 in the C&M module could, with more RAM added, be modified to perform the basic processing. The latter would require extensive hardware and software modifications to the C&M module.

APPENDIX A. THE BASIC OPERATING SYSTEM

A.1 INTRODUCTION

The following sections describe the basic operating system (BOS), which is used in the PDP11-compatible hosts on the COMSAT local network, called the distributed computer network (DCN). The BOS is a multiprogramming executive providing process and storage management, interprocess communications, input/output device control, and application program support. It operates with any PPD11 or LSI-11 model including at least 8K words of storage, an operator's console, and a communications device for connection to the DCN. For configurations including a disk, the system supports standard RT-11 operating system compilers, utilities, and user programs in either a stand-alone or network mode. Other features include support for multiple foreground and background jobs, which can utilize hardware memory management features to provide individual virtual address space.

The following describes the various BOS components and their operation. Also described are the various primitive functions and commands used to control network operation and the application programs. Previous reports and articles, listed in the references, describe the DCN as a whole.

The BOS architecture is based on multiple, independent, portable processes together with a hierarchial interprocess communications system. The system is designed to support applications in which the program need not necessarily be bound to a single processor and in which resources can be assigned dynamically as a function of requests. In terms of the process, the interface to all system services, including resource allocation, storage management, and interprocess communications, is independent of the

particular processor which implements these services. Services are provided by the supervisor, a version of which runs in every processor, and by a network of portable processes, copies of which are distributed throughout the network as determined by hardware resources and response time requirements.

In terms of the system, the BOS manages resources, including processor scheduling, memory allocation and input/output device operations. It supports the application program by providing the interprocess communications system, direct-access file system, and RT-11 emulator system. For the network, it provides communications services supporting both DCN network protocols and DARPA network protocols such as IP and TCP. The BOS can, however, be operated in a stand-alone mode for applications requiring fast pre-emptine multiplexing of cooperating independent programs. In this mode, the BOS provides services similar to the ELF and MOS systems developed for the PDP11 and LSI-11 by Bolt, Beranek, and Newman and by SRI International.

Communications devices supported in the present system include the DC11, DL11/DLV11, and DU11/DUV11 serial interfaces, DR11-C and DRV11 parallel interfaces, and special interfaces built by University College London (HDLC) and Stanford Research International (ARPA 1822). Direct-access storage devices supported include the RK11/RK05 and RL01 disk cartridge drives, RH11/RJS03 fixed-head disk drive, RX01, RX02, and AED 6200 flexible disk drives, and the Canberra 2020 magnetic tape cassette. The amount of storage necessary depends on the requirements of the application program and the particular input/output devices configured in the system. Any combination of application programs can be supported up to the limits of storage available.

In the simplest configuration, the BOS is a main-storage only system and does not support extended memory or direct-access devices. When configured with a direct-access device, the system

supports standard RT-11 linking, loading, and overlay operations with the additional support of multiple foreground jobs, each with a separate address space partition. Most programs designed to run in the single RT-11 foreground partition can run in any of these multiple BOS foreground jobs. When memory management hardware is available, multiple background jobs are supported, each of which can be assigned a separate virtual address space.

The structure of the BOS is modular and designed to adapt readily to different machine configurations. Only the supervisor module and supervisor data module require reassembly if the process configuration is changed. Only one set of tables in the supervisor data module requires modification when the input/output device configuration is changed. Although both the supervisor module and supervisor data module are required in every system, other system and application program modules are included only as necessary and require no change for the various machine configurations. At present, BOS load modules have been configured for several systems including the LSI-11/2/23, GT-40, and PDP-11/20/34/40/45/55, with storage sizes ranging from 8K to 124K and with varying numbers of communications interfaces, direct-access storage devices, and, in the case of the GT-40, a cathode-ray display.

The following sections contain a comprehensive description of the BOS as it now exists. The first part includes information useful for understanding the basic architecture of the BOS and its operation. The second part describes the structure of the supervisor and system processes, and the third includes detailed information on the structure of the user processes and the RT-11 interface.

A.2 PROCESSOR AND HOSTEL STRUCTURE

A BOS process includes procedure code, data storage, and an interface for communicating with other processes and components of the operating system. This environment is managed by the kernel (see Figure A-1), which includes mechanisms for input/output device interface, storage allocation, process scheduling, interprocess communications, and other similar functions. A hostel is an abstraction of an environment in which a set of processes can interact with the kernel and each other. The BOS includes those components of the kernel resident in the computer (the supervisor) and, in addition, certain processes to manage input/output devices, interhostel communications, and the file system.

The BOS is designed for the PDP11; therefore, the PPD-11 hardware itself is a hostel. The virtual operating system (VOS), another system constructed for the DCN, includes a multiprogrammed virtual machine capability in which PDP11 programs can operate, in particular, the BOS. Thus, a VOS virtual machine can be a hostel for the BOS. On the other hand, the VOS virtual machine does not support the memory mapping capability of the PDP-11/34/40/45/55/70, which is necessary for BOS operation. Thus, a VOS virtual machine cannot be a hostel for the VOS.

The environment viewed by a BOS process cannot bind the process to the particular hostel. Therefore, the interface between the process and all external objects accessable to it must be mediated by the kernel. This is implemented by the EMT instruction that passes control directly to the supervisor which, in turn, preserves the state of the process for later reentry. The supervisor is entirely event driven and strictly limited in processing time for any event, thus providing minimum latency for real-time device interrupts.

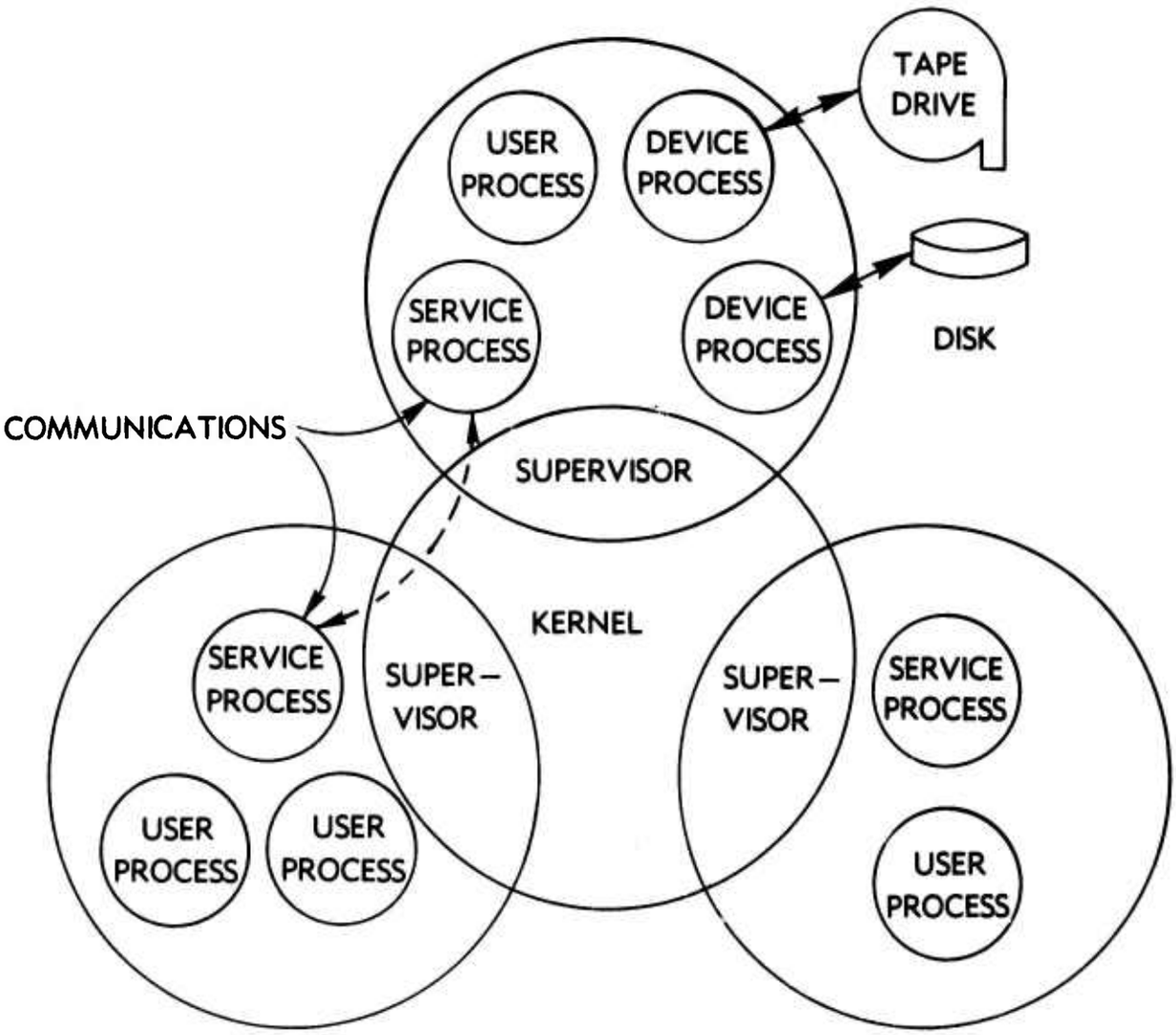


Figure A-1. Network Structure

Processes communicate with each other by messages, which can assume one of several forms. Intrahostel messages have a simple datagram-type format with reliable delivery assured. Interhostel messages have internet protocol IP-4 format, with reliable delivery assured by telecommunications protocol TCP-4 when necessary.

In all but exceptional cases, a process cannot inspect or modify a region of memory allocated to another process or the supervisor, hence the processes can be in the same or different hostels without any special action by the programmer. In some cases, a closely coupled set of processes are used to manage a network resource such as an operator terminal. In these cases, the processes involved may share a portion of their address space and may use a local message and semaphore protocol in the interest of efficiency. When communicating with processes not in the set, however, the standard interprocess message system is used. In other cases, a system service process implements a block-transfer operation between a direct-access device and a portion of memory allocated another process, again for efficiency. In these cases, both processes are constrained to reside in the same hostel, although possibly in different virtual address spaces. In still other cases, such as communicating with other networks such as SATNET or ARPANET, a service process implementing a network-specific protocol such as TCP is used, the interface to which is on a block transfer basis. For the above reasons, application processes cannot directly access files or protocol processes resident in another hostel.

A.3 SYSTEM STRUCTURE

The BOS structure consists of the supervisor, a set of user processes, and a set of system processes. The supervisor

consists of a number of routines to schedule processes, dispatch device interrupts, and coordinate interprocess communications and synchronization. User processes are designed to execute one or more application programs, depending on the particular system configuration. System processes include communications and disk device processes and protocol processes for communications with other hostels and networks.

The address space of the BOS is organized as shown in Figure A-2. In systems without memory management hardware, the memory segments are nonoverlapping. In systems including this hardware, the virtual space of each user process is mapped as required to real memory segments with the exception of the last 4K-work block of user space. This block is mapped for each process into the RT-11 emulator code segment and certain data areas used by the emulator. The supervisor, as well as the set of service processes, normally operates in kernel space without relocation, except for those service processes that manage block transfers between a device and a user process. In these processes, a 4K-word window in kernel space can be used to access the virtual memory of the user process. The supervisor automatically initializes the hardware relocation registers when switching from one user process to another.

The supervisor data module orders the segments shown in Figure A-2. In non-relocated systems, the RT-11 background region occurs first and can be used by only one user process. Instruction and read-only data areas for the supervisor and service processes occur after this region and are followed by instruction and read-only data areas for the RT-11 emulator and other resident modules used by the user processes. The latter areas are mapped into each user process within the last 4K-work block of virtual memory and are not used by the supervisor or service processes.

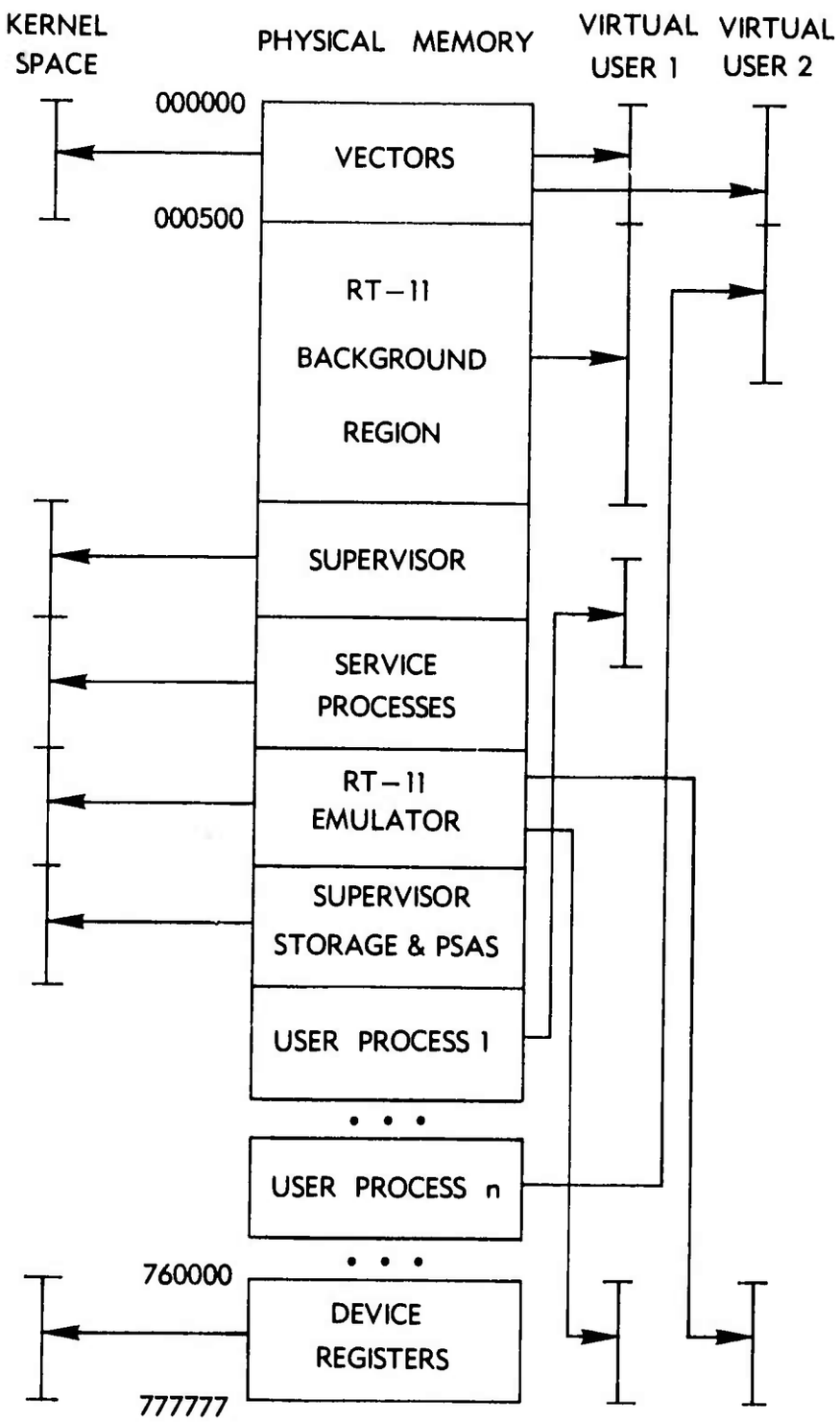


Figure A-2. BOS Address Space

Certain storage areas used by user and service processes (the PAR areas) must be accessed by the RT-11 emulator and other subroutines used by application programs. To provide access to these areas, they are grouped together in a single control section and mapped along with the RT-11 emulator in the last 4K-word block of virtual memory. Thus, these areas have two addresses: a real address (in kernel space) used by the device or service process, and a virtual address (in user space) used by the user processes.

The storage areas are followed by areas used for private stacks and other read/write data by each process in the system, including an area used by the supervisor for resource allocation and management. In non-relocated systems, the area assigned to each process includes the entire working memory allocated to that process; in relocated systems, this area includes only the data used by the supervisor for resource allocation and management - the working memory is allocated elsewhere. The supervisor stack is at the end of this area and is not normally included in the virtual memory of any user process.

In relocated systems, the remainder of the processor storage, up to 124K words, is available for allocation to user processes. Therefore, a number of RT-11 foreground or background jobs can be realized each with up to 28K words of virtual memory. In each of these jobs, the full 28K-word address space is available, since the other system components are either mapped into the kernel space or the last 4K-word block normally used for PPD-11 device registers.

In non-relocated systems, each of the user processes is assigned a nonoverlapping area for working memory, and all such areas must be contained in the 28K-word (30K words in some systems) address space of the machine. One of these processes can be assigned the background region described previously and can run

absolute load modules normally linked with the SAV file extension. The remainder are assigned foreground regions and can run relocatable load modules normally linked with the REL file extension. A special relocation scheme, not compatible with RT-11, allows suitably constructed position-independent load modules to be loaded and run in either the background or foreground regions.

User processes are designed to run programs conforming to the RT-11 operating system specifications, including most of the RT-11 system programs, such as the editors, compilers, linkers, and utility programs. Programs compiled under either BOS or RT-11 can also be run in a BOS user process. To support the RT-11 environment, a reentrant emulator package is included in the address space of every BOS user process. Certain operations such as direct-access device directory manipulations and data transfers are handled by dedicated service processes.

A.4 PROCESS STRUCTURE

Figure A-3 shows the anatomy of a typical BOS process. The procedure code, which has various control sections, is reentrant and can be shared by a number of processes. References within the procedure are, in general, position-independent. The read-only data, consisting of one or more control sections, are shared by all processes using the procedure. Reference between the procedure and data areas can be absolute, relative, or indexed, depending upon the program structure.

A transfer vector in the read-only data area is used by the supervisor during process initiation and during operation to send interrupts. As a programming convention, the routines invoked by interrupts are grouped together within the procedure and are distinct from other routines. This encourages compatibility

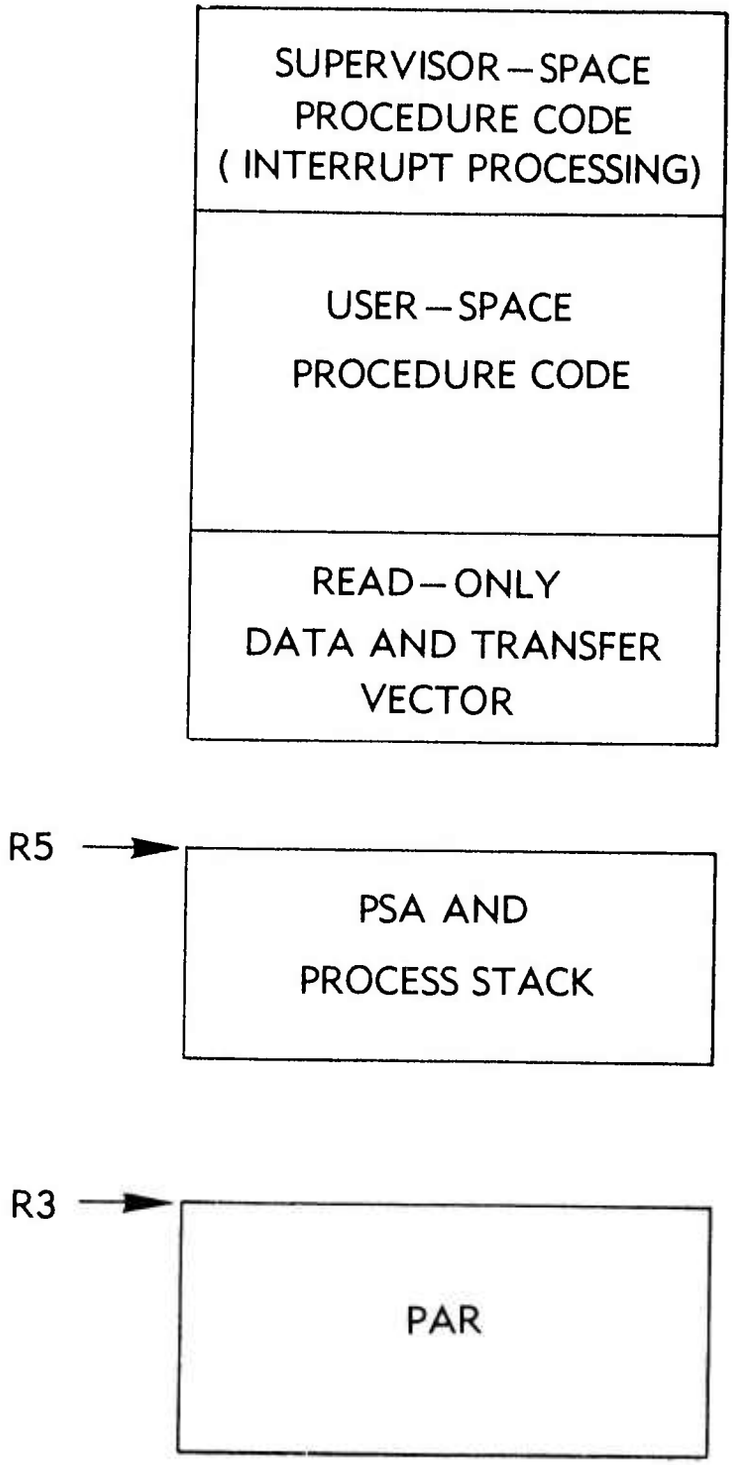


Figure A-3. Process Storage Layout

between the BOS and VOS systems, since address spaces used in interrupt and normal processing are not coincident in the VOS.

Each process has a distinct fixed size private storage area called the PSA. The PSA is used by the supervisor for scheduling and synchronization and by the process for its own local storage. By programming convention, general resister R5 is reserved as a pointer to this area. The PDP-11 architecture requires the use of general register R6 (usually called SP) as the CPU stack pointer. The process stack resides in the PSA. By programming convention, general register R4 is reserved as a pointer to the process stack area during interrupt processing only. During other processing, R4 can be used for scratch. The supervisor stack is used during the processing of interrupts.

The format of the PSA is derived from the supervisor and the process. The first portion is used by the supervisor for scheduling and message transmission functions, and the second is available for process use. The third and final section is reserved for the process stack. When the process is in the running state, the CPU stack pointer points to this area. When the process is suspended, the general registers are saved on the process stack. When the process is entered as the result of an interrupt, the supervisor sets base registers to point to the PSA and the block of general registers.

Initial parameters and other data are stored in the parameter area termed the PAR. By programming convention, general register R3 is reserved as a pointer to this area. In general, the PAR contains such data as the process name, device register, and interrupt vector location, and other data used to specify individual process parameters. In some cases, a synchronized network of processes is used to control a particular device. For example, a network of processes controlling a terminal consists of keyboard and printer processes. In these cases, the PAR

is common to both processes and can be used for the exchange of information.

In addition to the base registers assigned to the PSA and PAR, most device processes used general register R2 as a pointer to the register set assigned to the device (the DEV area). This provides a convenient reference and more compact coding than computing offsets from the device register address stored in the PAR. During interrupt processing, the remaining general registers R0 and R1 and the supervisor stack can be used as scratch; during other processing, general registers, R0, R1 and R4, and the process stack can be used as scratch. User application programs supported by the RT-11 emulator can use all registers and storage areas.

A.5 PROCESS OPERATION

A process has four states (Figure A-4); running, pending, blocked, and waiting. A process in the running state controls the CPU. The hardware registers of the CPU stack pointer indicate the stack portion of the PSA. A process in the pending state is waiting to gain control of the CPU. Its PSA is linked on a priority queue maintained by the supervisor. When the process advances to the head of its queue and all higher priority queues are empty, it will be placed in the running state. A process in the blocked state is waiting for a semaphore, the interval timer, or a message. When the semaphore is unblocked, the interval timer overflows, or a message arrives, the process will enter the pending state and, eventually, the running state. A process in the waiting state is waiting for an asychronous interrupt such as may be generated by a hardware interrupt. When the interrupt arrives, the process can enter the pending state.

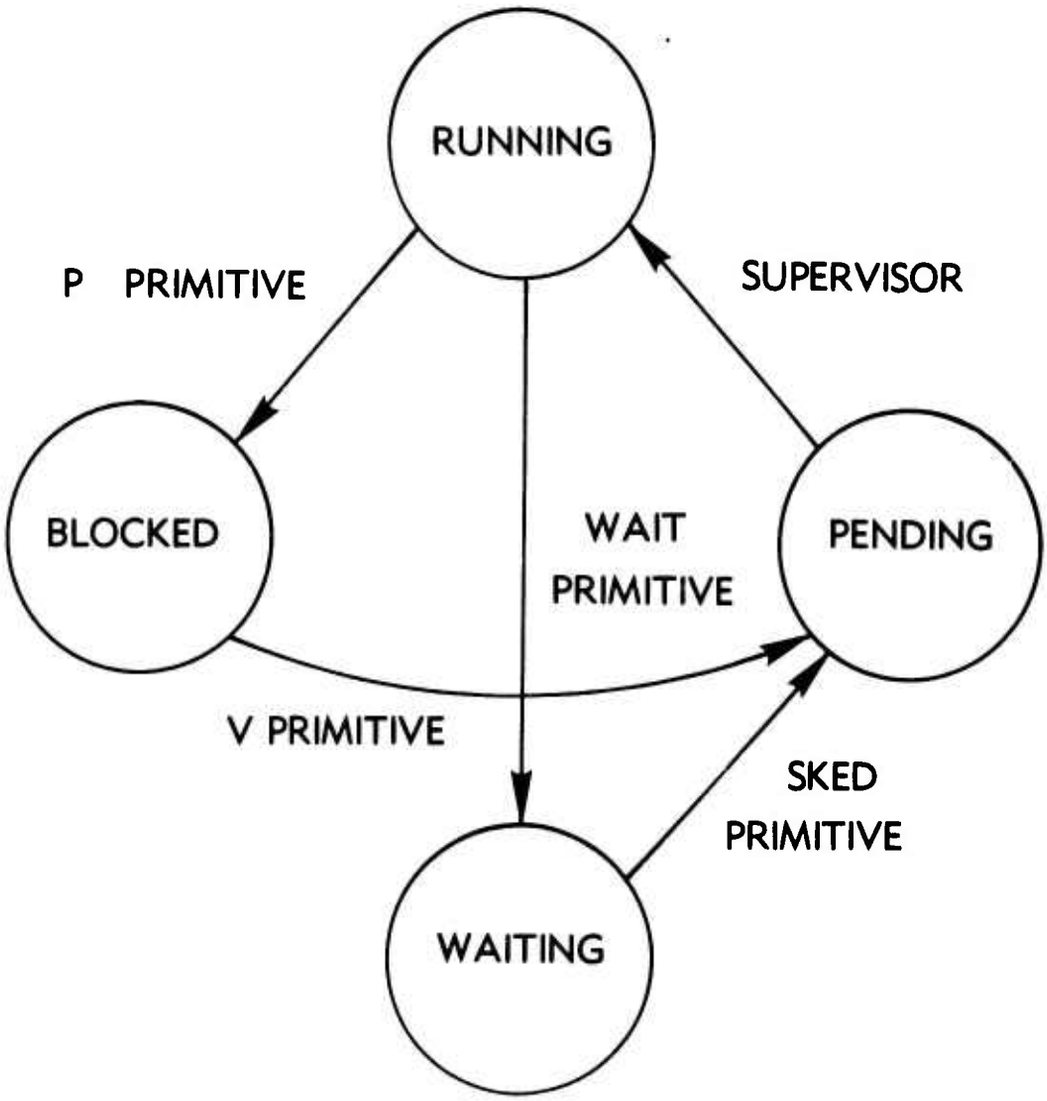


Figure A-4. Process States

The system itself can also be in one of three states: the process state, in which some process controls the CPU and the CPU stack pointer points to the stack for that process; the supervisor state, in which all processes are suspended and the CPU stack pointer points to the supervisor stack; and the wait state, in which the computer is stopped on a WAIT instruction and must receive a hardware interrupt to resume processing.

Processes in the BOS are not time-sliced. Accordingly, a process can be changed from the running state only by a supervisor-call operation which places it in the pending, blocked, or waiting states or by an interrupt which activates a higher priority process. In the latter case, the interrupt places the running process in the pending state. The process enters the pending state from the blocked state because of an explicit call (semaphore operation or message transmission operation) on the part of another process. The process enters the pending state from the waiting state because of an asynchronous interrupt generated by a hardware interrupt of an explicit call from another process.

A process can gain control of CPU in two ways: it can be activated by the supervisor in the running state or called by the supervisor in the supervisor state. In the former case, the process completely controls the CPU and can enter other states at will. For system processes operating in kernel space, the CPU stack pointer points to the process stack in the PSA. For user processes operating in user space, the CPU stack pointer is set as specified by the application program. When called directly by the supervisor in kernel space because of a synchronous or asynchronous interrupt, the general registers are set as required by the calling process or supervisor and the process must return, as if called as a subroutine, to the supervisor. In this case, the CPU stack pointer points to the supervisor stack, and the contents of the general registers at the time the process was

suspended are stored on its process stack in the PSA. Base registers R4 and R5 will be set to the register area in the PSA and the PSA pointer, respectively, before the call is made.

A call in the supervisor state can occur because of either an interrupt during the running state of the process itself or a call by another process or by the supervisor. In the former case, termed a synchronous interrupt, the interrupt can be due to an error (e.g., address exception) or an extended supervisor primitive. In the latter case, termed an asynchronous interrupt, the interrupt is due to a special supervisor call from another process. In either case, the transfer vector (Figure A-3) is used to vector the interrupt to the appropriate subroutine as a function of the type of interrupt. An indefinite number of interrupts can be defined using appropriate codes and transfer vector entries.

An extended interrupt is initiated by an EMT instruction and usually used to enter the supervisor state to initiate a device operation and enter the waiting state pending completion. Since hardware interrupts are disabled in the supervisor state, race conditions are avoided. An asynchronous interrupt is generated at the conclusion of a device operation because of a hardware interrupt from the device. The activiated subroutine saves information, such as the character read and device status, in the register area in the PSA so that it is available when the process becomes active. Upon completion of the data transfer, the subroutine places the process in the pending state and exits to the supervisor. Thus, complete control over state transitions and device operations is possible using a minimum of unique code.

The asynchronous interrupt is also used to signal the temporary suspension of a process because of an exceptional condition, such as a defined keystroke at a user terminal or a transient fault of a dependent process. The asynchronous interrupt is used in conjunction with the attention message to effect

the behavior expected by a user who interrupts an errant program or stops printing.

A.6 INTERPROCESS COMMUNICATIONS AND SYNCHRONIZATION

The BOS architecture requires that processes cannot communicate using shared storage. Instead, a highly developed interprocess communications and synchronization facility is required. Functional control over message formatting, flow control, and routing is provided by the supervisor, which includes primitives for sending messages on a byte-by-byte basis. A system of semaphores is used to control the multiplexing of messages from several senders to a single receiver. The semaphore facility is used for other purposes as well, including synchronization of system processes within the hostel.

A message, consisting of a string of bytes preceded by a header, is sent by a process to a connector called a port. One or more ports can be allocated to a process on request. A port is always allocated to a single process, although any process can send to it. Every process is allocated a particular port, called the process port, which is used to communicate system messages to the process. The name of the process is also the name of this port. User processes have another port, termed the reply port, which is used to transact operations with other processes while subsequent system messages remain queued on the process port. Additional ports can be allocated by a process for special situations. Ports are identified by a unique integer called the PID.

Message transmission is on a byte-by-byte basis in one of two formats, word format or record format. In word format, the message is complete in a single word (two bytes), and successive words in a received message may contain messages from different processes. This format is used primarily by hostel

service processes. In the record format, the message may extend over one or more bytes and is preceded by a header consisting of a status byte, a count of the number of bytes in the message, the PID to which the message is sent, and a PID to which a reply can be sent.

Although the supervisor manages message transmission on a byte-to-byte basis throughout the system, each process is responsible for formatting its messages to a system standard and to obeying a standard transmission protocol for flow control. A standard interpretation of status bits, command codes, and other protocols is followed by all processes.

A set of modifier bits is included in the status byte of each message along with the command code. These bits control certain auxilliary functions affecting code conversion and transmission status. They include the binary bit and three status bits: the error, end-of-file, and attention bits.

The binary bit, which is uninterpreted at the message transmission level, has meaning only to the sending and receiving processes and is usually reserved for code conversion and media format functions. The error bit is normally used when a transmission or format error occurs in the processing or interpretation of a message. This bit is interpreted; at the discretion of the sending and receiving processes, which may ignore the error, discard the message, or request a retransmission. For instance, some processes set the error bit in the acknowledgment transmitted in reply to a write message to indicate that the process detected an error.

A standard interpretation of the end-of-file bit by various processes controls the blocking of messages into files and provides for multiplexing of user processes. When a process does not have any more data to transmit, it signals the end-of-file condition. This can also be indicated by a keyboard input

process because of a specified key. When a process receives a message with the end-of-file bit set, it normally does not issue any more read messages requesting data. A process which receives a write or control message can transmit the end-of-file bit in its acknowledgment.

The timeout bit is set when an acknowledgment to a read, write, or control message has not been received within a specified interval. This feature can be used for system error recovery, polling, and broadcasting operations, or any other activity requiring a specified delay interval.

The attention bit is used in conjunction with the asynchronous interrupt to provide the attention-interrupt feature required in interactive time-sharing systems. The attention interrupt is intended to temporarily disable data transmission to a device or to suspend computation by a user process. Although it can be generated by any process at any time, it is most often initiated by a keyboard input process in response to a special attention key.

Three message formats are presently used in the BOS: stream, record, and packet (Figure A-5). Stream format consists of the message header followed by a maximum of 255 bytes of data and is used primarily between control (e.g., operator console) and user processes. Record format is used to exchange storage pointers, command, and state information between user and direct-access device processes. The structure of these messages is similar to that of the RT-11 argument block. Packet format is used to exchange packet pointers, control, and status information at various stages in the interhostel communications system, including the internet process and its dependent protocol modules.

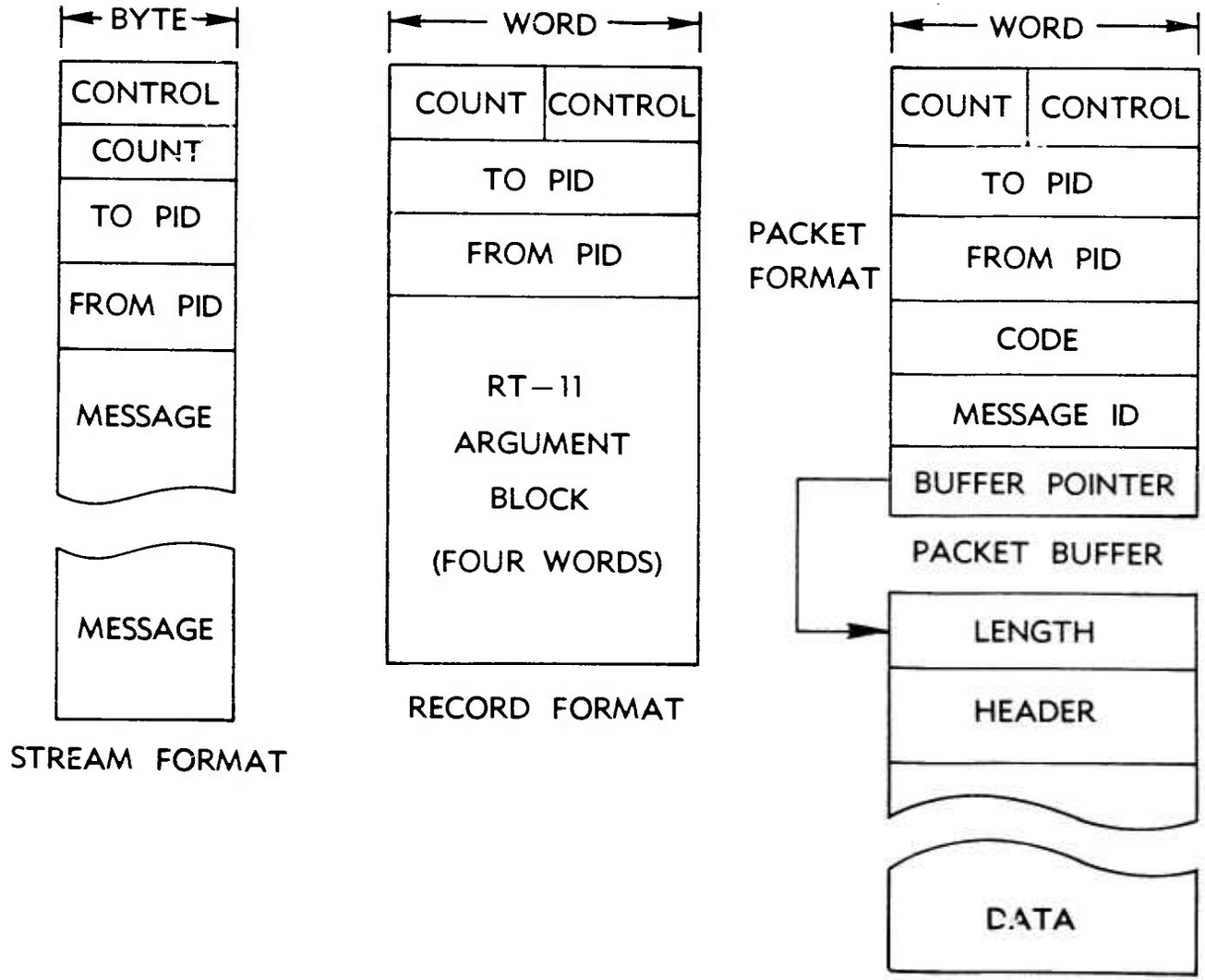


Figure A-5. Message Formats

A.7 SEMAPHORES

A system of semaphores is provided to synchronize processes. Semaphores are implemented as a system of queues, one queue for each semaphore. Each semaphore has an associated value which can be accessed only through specified supervisor primitives. A semaphore can be accessed only by processes in the same hostel as the semaphore itself.

There are two possible operations on semaphores: the P and the V operation. The P operation causes a conditional wait. If the semaphore queue is nonempty, the process is blocked and entered at the end of the queue. Otherwise, the semaphore is decremented. If the result is negative, the process is blocked and entered at the end of the queue. The V operation causes the first process on the queue, if any, to be placed in the pending state. If no process is on the queue, the semaphore value is incremented.

The P operation can be performed only by a process in the running state. On the other hand, a V operation can be performed by a system component in any state (possibly because of an asynchronous interrupt). Sometimes a process must be blocked in the supervisor state. The V operation can also be accomplished in this state, with a condition code indicating whether the process entered the blocked state as a result. In these cases, the PC can be backed up and a bit set so that the supervisor state code can be repeated when the process is unblocked.

A.8 THE SUPERVISOR

The supervisor includes the mechanisms for process switching, scheduling and dispatching, message transmission,

process initiation and termination, interrupt processing, and storage allocation. The tables used to control these functions are accessed only by the supervisor and not by the various processes which depend upon the supervisor for service. Entry to the supervisor occurs only as a result of an emulator trap (EMT) instruction or hardware interrupt.

The structure of the supervisor, as shown in Figure A-6, consists of a set of interrupt response routines which save the context of the machine on a process or supervisor stack and then cause a number of other subroutines to be called, depending upon the nature of the interrupt and the processing indicated.

A.9 CPU SCHEDULING

Process scheduling is determined using the set of tables shown schematically in Figure A-5. Each process is identified by an index stored in its PSA. This index, called the process ID, identifies an entry in the PSA pointer table (CPUPSA), which in turn contains a pointer to the PSA. The entries in the PSA pointer table are in one-to-one correspondence with the CPU queue table (CPUQUE), which is used to store links (process IDs) on the various system queues.

There are a number (currently five) of CPU queues corresponding to the number of different priorities assigned to the various processes in the system. The CPU queue-head table (CPUHED) contains an entry for each of these queues giving the process ID of the first and last process on the queue. If any of these queues are empty, the corresponding CPU queue-head table entry is zero. The following is a typical assignment of priorities in the system:

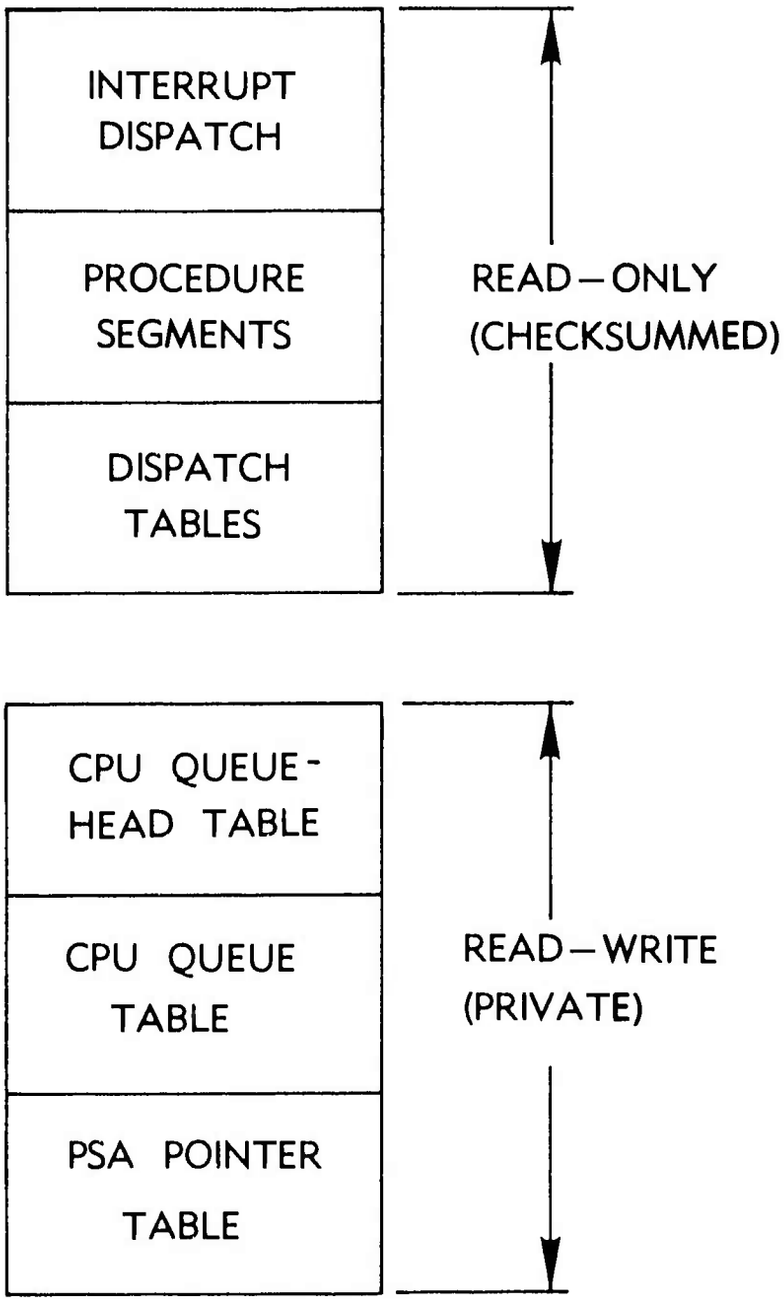


Figure A-6. Supervisor Structure

Priority	Processing Type
0	not used
1	user processes
2	service processes (e.g., internet process)
3	asynchronous (nonoverrunable) devices
4	synchronous (overrunable) devices

When a process is to be dispatched from the pending to the running state, the supervisor scans the CPU queue-head table starting at the entry corresponding to the highest priority. At the first nonzero entry, the supervisor removes the first process in the indicated CPU queue table entry and saves the PSA pointer from the corresponding PSA table entry in a special location (ACTIVE). The supervisor then loads the general registers from the PSA indicated and activates the process using a return-from-interrupt (RTI) instruction.

If all of the CPU queues are empty, the supervisor enables interrupts (by setting the CPU priority level to zero) and stalls on a WAIT instruction. No further activity will transpire until a device interrupt occurs. The interrupt will be serviced as usual (in the supervisor state), which results in a return to the first instruction following WAIT. Then, the supervisor scans the CPU queue-head table for a nonempty entry and proceeds as before.

Difficulties were encountered during the development of the BOS due to the differing interrupt processing conventions in the various PDP11 models and, in particular, due to the attempt to avoid an interrupt between instructions which set the CPU priority to zero and the WAIT instructions. If an interrupt were accepted at that point, processing would stop on the WAIT instruction and the CPU queues would not be scanned. Since it proved

impossible to find a single instruction sequence which would work in all models, it was decided to allow the interrupt and then check the program counter (PC) stored in the PSA just before returning from an interrupt. If the instruction about to be executed is WAIT, the program counter is advanced to the next instruction before the execution of RTI instruction. The architecture of the LSI-11, unlike other PDP11 models, does not provide capabilities for accessing the processor status word (PS), except through special instructions; therefore, a special sequence is constructed using the MFPS and MTPS instructions.

A.10 INTERRUPT PROCESSING

Entry to the supervisor occurs only because of an interrupt. When this occurs, the CPU priority is set to disable device interrupts and the general register contents stored on the current CPU stack. If the interrupt occurred from the process state, the registers are saved on the process stack in the PSA; if it occurred from the supervisor state, the registers are saved on the supervisor stack. If the interrupt is from the process state, the stack pointer value, after storing the registers, is saved in the PSA and the CPU stack pointer is initialized to the beginning of the supervisor stack.

The interrupt may be directed to the interrupted process (synchronous) or to another process (asynchronous). In either case, the general registers are set before entry to the interrupt routine as follows:

R0	not changed (can be used to pass parameters)
R1	port identification (PID)
R2	loaded from R2 in PSA (usually DEV base register)

R3 loaded from R3 in PSA (usually PAR base register)
R4 REG base register (usually scratch in run state)
R5 PSA base register
SP supervisor stack pointer

The interrupt routines can use the base registers so that computed results can be inserted into the registers when it is next activated by the supervisor.

When a process is activated, the PSA pointer is stored in a special location (ACTIVE). When the process is interrupted, this location is used to retrieve the PSA pointer, rather than R5. Thus, it is not necessary for the process to preserve any registers if they are not needed for PSA addressability. The saved PSA pointer is also used to avoid unnecessary CPU queue head table scans when the process is reactivated and no other processes of higher priority have been scheduled.

Interrupt processing depends on the type of interrupt. If due to an EMT, usually termed a supervisor call, the code in the EMT instruction is extracted and a branch is made via a table which contains an entry for each defined code. If the interrupt is because of a power-fail/restart sequence, an immediate branch is taken to the initialization procedure. If it is due to a device interrupt, an immediate branch is taken to the device process procedure. In all other cases, if the interrupt occurred from the process state, a synchronous interrupt is generated; if from the supervisor state, the system halts in an error condition.

Interrupt processing is facilitated by a transfer vector, which is part of the program module associated with the process. The location of this table is given by an entry in the PSA. The first entry in the transfer vector contains the location of the error interrupt routine, while the second contains the location of the asynchronous interrupt routine. Additional entries contain the locations of extended supervisor primitive interrupt routines.

A process interrupt routine is called immediately after the registers are stored on the process or system stack as described previously. These routines operate in the supervisor state and return via a return-from-subroutine (RTS) instruction. A number of supervisor subroutines can be useful in this state. One is the WAIT subroutine, which places the running process in the wait state. Another is the SKED subroutine, which places the waiting process in the pending state. In addition, since the structure of the supervisor is recursive, a supervisor call can be made from supervisor state; however, this is avoided when possible, due to the additional overhead in register save/restore operations.

A.11 SYSTEM INITIALIZATION

When the system is initialized, a checksum of all the invariant procedure and data areas is computed and compared with the checksum computed on the previous initialization. A discrepancy indicates that one or more words in these areas have been changed since the previous initialization. This has proved an effective debugging tool because, as long as the integrity of the storage, which is indicated by a correct checksum, is maintained, the system is serially reusable and can be reinitialized at any time.

Following the checksum computation, all of the available storage area from the beginning of the supervisor data area to the end of the supervisor stack area is reset to zero. This ensures a stable state of storage to aid in the isolation of apparent random errors.

Next, the supervisor initiates the system and user processes, as indicated by the configuration table. This involves

allocating the PSA for each process in turn, starting at the end of the supervisor data area, and scheduling that process on the appropriate CPU queue. Information used to determine the priority, PSA length, and initial starting address is derived from the initialization portion of the transfer vector. The general registers are set as follows when the PSA is initialized:

R0	interrupt vector pointer (from PAR)
R1	process port PID
R2	DEV area pointer (from PAR)
R3	PAR pointer
R4	PAR pointer
R5	PSA pointer
SP	process stack pointer

When all of the PSA areas have been allocated, a specified number of packet buffers for use by the internet system are allocated and the remainder of storage to the supervisor stack area is initialized for message buffers. Following this, the initialization routine branches to the scheduler to activate the first process. Each process contains a small routine to perform initialization functions such as allocation of ports and interrupt vectors. Both of these functions are performed by the supervisor call primitives using information supplied in the PAR.

Following the operations described above, the system is in an operational state and continues to run indefinitely. In general, a CPU interrupt (trap to a CPU interrupt vector) is isolated to the process in which it occurs and causes a synchronous interrupt to occur to an address given in the transfer vector. If no such address is given, the supervisor halts at a designated location. If an error is detected in supervisor state, the supervisor halts at another location. The only other supervisor halt

occurs when the message buffer area is exhausted. After any of these halts, it is almost always necessary to reinitialize the system.

A.12 SUPERVISOR PRIMITIVES

A supervisor primitive is a subroutine that is usually invoked because of a supervisor call (EMT instruction). Most of these primitives can also be called from interrupt routines. A partial list of primitives presently available in the BOS is given in the following with a short informal explanation of the function of each.

.INIT

Branch to the initialization procedure. This has the same effect as if machine power were just turned on, or if the system were restarted at location 000000.

.PRIO

Reset the priority for the calling process to the value contained in R0. If R0 contains zero, do not change the priority. In either case, return the old priority in R0. This primitive can be called only in the running state.

.WAIT

Enter the wait state. This primitive can be called only in the running state.

.SKED

Enter the pending state. This primitive can be called only in the supervisor state when the process to be scheduled is in the waiting state.

.SETV

Create a device interrupt vector. Vectors are allocated dynamically in a table intended for this purpose. The location of the vector is given by R0. The interrupt codes that determine which transfer vector entry to use are given in R1.

A.13 INTERVAL-TIMER PRIMITIVES

An interval timer facility is included in all systems that have the appropriate hardware. Hardware devices supported include the KW11-L line frequency clock, KW11-P programmable real-time clock, and the event timer which is part of the LSI-11 microprocessor. The interval timer facility operates by suspending the process for a specified number of clock ticks, or causing an asynchronous interrupt, as determined by the supervisor primitive. Another supervisor primitive can be used to abort the interval and reschedule the process.

The following is a list of supervisor primitives used to control the interval timer facility:

.STIM

Start the process timer to run for the number of clock ticks specified in R0. Upon expiration of the timer, initiate an asynchronous interrupt to the process.

.DLAY

Suspend the calling process for the number of clock ticks specified in R0.

.CTIM

Remove a pending timer operation from the system queues. Return the clock ticks (.DLAY or .STIM) remaining in R0. This primitive is usually called in the supervisor state because of a device interrupt.

A.14 SEMAPHORE PRIMITIVES

Semaphores are associated with ports for flow control and message interlocking. They can be used separately or in conjunction with the message transmission primitives. Each semaphore is identified by the port ID. The same mechanism used to allocate a port (.GPRT) can be used to allocate a semaphore.

The following operations can be performed on semaphores:

.PSEM

Perform a P operation on the semaphore with the PID contained in R1. If the primitive is called in the process state, the process will enter the blocked state or resume processing in the running state as determined by the value of the semaphore.

.VSEM

Perform a V operation on the semaphore with the PID contained in R1. This primitive can be called in any state and, in particular, as the result of an interrupt.

A.15 MESSAGE TRANSMISSION PRIMITIVES

Messages are exchanged between ports identified by PID and can be transmitted in two formats: word or record. Word-format messages are intended for functions such as terminal echo and interprocess synchronization. A word format message is sent by a process using the .send primitive and received using the .RECV or .RECA primitives. Synchronization and buffering procedures are automatic; up to 255 word-format messages can be outstanding before overflow.

Record-format messages are used when more than a single 16-bit word is necessary. A header precedes every record-format message and has following form:

Byte	Function
0	status
1	byte count
2,3	send PID
4,5	rely PID
6	message text

The byte count field of the header applies only to the message text; a value of zero implies a null message. The status byte is coded into two fields: message (bits 0-1) and status (bits 3-6). Bit 2 is used as a modifier bit and usually interpreted to indicate a binary (non-ASCII) message. The higher-order

bit (bit 7) is not used. A record-format message is sent by the following sequence of primitives:

```
EMT     .PUTH     ;BUILD HEADER
EMT     .PUTB     ;SEND
...               ;BYTES
EMT     .PUTB     ;OF MESSAGE
EMT     .CLSE     ;TERMINATE MESSAGE; SET
                   STATUS
```

The message text may consist of a maximum of 255 bytes. The byte count field of the header is constructed automatically. A record-format message is received by the following sequence:

```
EMT     .RECV     ;(OR .RECA) WAIT FOR
                   MESSAGE
EMT     .GETH     ;FETCH HEADER
EMT     .GETB     ;FETCH
...               ;BYTES
                  ;BYTES
EMT     .GETB     ;OF MESSAGE
```

The operation of .RECV or .RECA with record-format messages is the same as that with word-format messages, except that the word returned is the first two bytes of the header. The low-order byte is the status byte; the higher-order byte is the byte count. Note that processes must have prior agreement as to whether a port will be used for word- or byte-format messages.

When it is significant, the condition code (CC) is set upon return from any of the message transmission primitives to indicate the status of the port and the results of the operations performed:

CC	Interpretation
C	invalid PID
V	buffer overflow or nonexistent port
N	timeout
Z	empty record

The following lists the message transmission primitives and gives a short description of their functions:

.GPRT

Allocate a port to the process issuing this primitive. The PID is returned in R1.

.RECV

Wait for the next input message to arrive on the PID contained in R1. If no message is received after the number of clock ticks contained in R0, return with the N bit set in the CC. Otherwise, return the first word (two bytes) in R0.

.RECA

Wait for the next input message to arrive on any port allocated to the process. If no message is received after the number of clock ticks contained in R0, return with the N bit set in the CC. Otherwise, return the first word (two bytes) in R0 and the PID in R1.

.SEND

Send the word contained in R0 to the PID contained in R1.

.PUTH

Construct a header for the next output message to be sent to the send PID contained in R1 with a rely PID in R0.

.GETH

Get the header for the next input message. The send PID is returned in R0 and the relay PID in R1.

.PUTB

Put the byte contained in R0 in the buffer.

.GETB

Return in R0 the next byte from the buffer.

.CLSE

Terminate the current output message and set the status byte as the value contained in R0.

.TPRT

Set the condition code corresponding to the status of the port whose PID is contained in R1. Return the number of messages queued on the port in R0.

.BKSP

Delete a byte from the current output message. The next byte should be transmitted with a .PUTB.

.CNCL

Delete the entire current output message. The next message should begin with a .PUTH.

.ASYN

Generate an asynchronous interrupt to the port whose PID is contained in R1. The asynchronous interrupt routine of that process is called with register R0 containing the PID.

A.16 EXTENDED SUPERVISOR PRIMITIVES

In many cases, some special process dependent operations must be performed in the supervisor state, such as the manipulation of a device register or the entrance to another state. A number of routines, termed extended supervisor primitives, can be called by using EMT instructions with codes assigned in certain ranges. EMT instructions with codes outside these ranges are interpreted according to RT-11 conventions for both Version-1 and Version-2 programmed requests. The location of the routine assigned to each code is given in the third and succeeding entries in the transfer vector.

This mechanism is also used to transmit hardware device interrupts to a device process. A hardware interrupt vector is

allocated by the .SETV primitive, which specified the location of the vector, the PID on which the interrupt is to be received, and the number of the entry in the transfer vector which contains the location of the interrupt routine.

APPENDIX B. DESCRIPTIONS OF ADDITIONAL SOFTWARE ELEMENTS

B.1 COMMAND LANGUAGE INTERPRETER

The command language interpreter (CLI) interprets and executes typical operator commands to initiate and control system operations. It is structured as a reentrant application program which can be made resident and shared by all user processes or dynamically loaded in the storage partition of each process separately, depending on a choice made a system-link time. The CLI is entered from a user application program via the .EXIT request or as the result of a program interrupt, system malfunction, or control-C interrupt.

CLI commands consist of a command name followed by a list of arguments separated by one or more spaces. Only the first three characters of the command or argument name are significant. Numeric arguments are converted from decimal or octal representation to internal form with a precision of 16 bits.

The CLI is organized into segments associated with the different process types in the system. For instance, the segment associated with the user process (foreground or background) supports commands for general purpose program initiation and debugging, while those segments associated with device processes support special purpose functions unique to each device. The SET command is used to specify the segment to which the following command applies, with the default segment assumed to be that associated with the user process. For instance, the command

```
SET TTØ MARGIN 8Ø
```

specifies the segment associated with the device TTØ (operator's terminal) and the command MARGIN 8Ø sets the right margin to the 80th column.

The following is the list of commands supported at the present time, together with a brief description of their functions.

DISPLAY arg1 arg2

Arguments:	arg1 = starting location (octal), arg2 = number of words (octal) (default one).
Function:	Display contents of memory in octal format. Each line consists of the address of the first word followed by the contents of the 8-word block beginning at that address.
Notes:	An attempt to access nonexistent memory results in a diagnostic message. Attempts to access a word in a region of the application program which is overlaid by the CLI rarely give meaningful results.

ALTER arg1 arg2...

Arguments:	arg1 = starting location (octal), arg2 = values (octal).
Function:	Alter contents of memory starting at arg1 to contain values arg2,....
Notes:	See notes under DISPLAY command.

RESET

Arguments:	None.
Function:	Reinitialize system.
Notes:	This a stong command equivalent to starting the system at location zero. It should be used with discretion.

PSA arg

Arguments:	arg = port ID of process (octal).
Function:	Display the process save area (PSA) with the port ID arg.
Notes:	See notes under DISPLAY command. Port IDs for the various processes can be determined with the SHOW command.

TEST arg

Arguments:	arg = repeat count (decimal).
Function:	Print test line arg number of times.
Notes:	Test line consists of 64 ASC11 characters with codes 040-137 (octal).

SET arg text

Arguments:	arg = process name, text = command.
Function:	Interpret and execute CLI command text with respect to the CLI segment associated with the arg process.
Notes:	This command is recursive, as might be expected, should the use of that feature be interesting or even useful.

ON arg

OFF arg

Argument:	arg = switch values (octal).
Function:	Turn on/off option switch bits.
Notes:	This command is normally used in conjunction with the SET command to enable/disable device features, etc.

SHOW arg

Arguments:	arg = process name, default all devices.
Function:	Display data pertient to process arg, or all processess if default. These data include process name, port address, type, max-blocks, interrupt vector address (if relevant), device register address (if relevant), and option switches. The format of all numeric data is octal.

DEVICE arg

Arguments:	arg = logical device number (octal), default all devices.
Function:	Display portions of the logical device table for this process. These data include for each entry logical device name, actual process name, port address, type and max. blocks.
Notes:	Unassigned entries are printed with *** in place of the name.

ASF arg1 arg2

Arguments: arg1 = LDN or logical device name, arg2 = actual process name.

Function: Assign arg2 as the process implicitly referenced by an application program request to device arg1.

RUN arg Load arg XCTL arg

Arguments: arg = file specification (RT-11 format).

Functions: RUN loads the file specified into memory and begins execution.

LOAD loads the file but does not begin execution so that modifications can be made.

Neither of these commands allows the loaded file to overlap the CLI, which is located in the memory area below the monitor communications area. The operation of XCTL is identical to RUN, but the loaded file can overlap the CLI, which then must be read from the disk when the loaded program terminates.

START arg

Arguments: arg = starting address.

Function: start execution of a program previously loaded by the load command.

B.2 RT-11 EMULATOR

The DCN kernel provides a set of features which simulate a standard RT-11 operating system environment. These features, collectively called the RT-11 emulator, allow system and application programs to operate in their native environment yet communicate with other systems using the protocol modules described in this report. In general, any RT-11 program that does not directly control a communication or storage device and does not depend on certain operating system features of the RT-11 can run without change with the RT-11 emulator. This includes all of the RT-11 compilers, utilities, and application programs provided with RT-11 release 3B and 4.0 together with most programs provided with earlier releases.

The RT-11 emulator operates similar to the RT-11 foreground background monitor in that a single absolute (non-relocated) background job can be run simultaneously with a relocated foreground job. In the emulated system, however, more than one foreground job can be run simultaneously with the background job, although the memory partition for each job is fixed and must be declared at system configuration time. Each of these jobs runs as a separate process under the BOS operating system at a priority specified by the user. Within its memory partition, the job is free to manage resources as required, including use of the memory block between 34-57 (octal) normally used by RT-11 programs to communicate with the monitor. Each job is assigned such a separate block, termed the cache, which is automatically saved and restored as required by the emulator.

Programs that run in the background process have exclusive use of its cache together with the block of memory beginning at location 500 and extending up to the bottom of the area used by the monitor communications area. This area is used

by the emulator in a fashion similar to but not identical to the RT-11 system. Programs that run in a foreground process have exclusive use of its cache together with a block of memory specified at configuration. Each foreground process has its own monitor communications area and operates as if it were in its own distinct RT-11 system. All processes, including the background process and the collection of foreground processes, are assumed to be well-behaved and are not allowed to alter the memory contents outside of their assigned areas, with certain exceptions as indicated.

The following assumes that the reader is familiar with RT-11 operating system concepts and facilities, including programmed requests, files and directories, and resource allocation. All interrupts, including programmed requests (EMTs) and traps, are intercepted by the emulator and processed or reflected back to the job as appropriate. Both the user service routine (USR) and the command string interpreter (CSI) are resident, reentrant and shared by all processes. The CLI, which is the BOS analog of the RT-11 keyboard monitor (KMON), can either be shared or dynamically loaded separately in each process on demand, as determined at configuration. The resident monitor (RMON) is represented by the emulator and other components of the BOS system. All devices are resident and do not require installing or loading.

Neither the background or any of the foreground processes are swapped; that is, there are no features for writing a portion of a job to a swap file, overlaying it with a system component, then restoring that portion later. However, normal RT-11 overlay operations are supported in both the foreground and background processes and, in the case of multiple background processes, separate individually relocated overlay files are maintained. In addition, standard RT-11 program loading mechanisms have been extended to allow specially constructed system programs to be loaded and overlayed to appear transparent to ordinary RT-11 programs. These extensions are used by the DCN/RT-11 TELNET server and foreground loader among others.

The emulator supports a file system identical to RT-11 in directory structure and file formats. Each disk drive controller is supported by a device driver process, which communicates with other processes by messages. The emulator maintains a 1-segment cache of a device directory and communicates with the user and device processes by messages. Files may be opened, closed, and accessed indiscriminately by any process, and no explicit mechanisms are provided for synchronization other than to ensure that only the rpocess that opens a file with an .ENTER programmed request can close that file. This situation requires care when performing certain system housekeeping functions, such as squeezing unallocated holes from a volume, when access by only a single job can be permitted.

User processes, both foreground and background, communicate with each other and with other BOS processes by messages. For the file system, appropriately formulated messages are automatically generated by the emulator. A disk transfer, for example, is initiated by a message carrying information similar to the RT-11 argument block for a .READ or .WRITE programmed request. While the message is being processed by the device and the data transfer (DMA) is occurring, the requesting user process can continue processing. The device process returns a message when the transfer is complete. The .WAIT programmed request can be used to wait for this message, or the .READW or .WRITW programmed request can be used before returning to the user program. The .READC and .WRITC programmed requests and the use of completion routines are not supported. It is possible, using a special programmed request (.EVENT), to wait for a number of concurrent input/output operations on different channels simultaneously, an operation required of protocol modules such as FTP.

A BOS user process is normally regulated by the control process. Control processes are most often represented by terminal processes, which interface operator terminals with the system. A

terminal process assembles characters into messages, performs line-editing and page formatting operations, and controls the terminal device. Using appropriate control function sequences, the terminal operator can switch among a number of user processes, generate a data stream for logging (script recording), and link to other terminals. Messages sent by one or more terminal processes to a user process can be read using the RT-11 .TTYIN and .GTLIN programmed requests. Output generated by the .TTYOU and .PRINT programmed requests is sent to a terminal (or other) process as specified by the user using the ASG command. In terms of the user program, a TELNET connection via TCP-4 appears as a standard terminal process and is transparent to the application program.

B.3 FOREGROUND LOADER

The foreground loader (FRUN) processes RT-11 files designed to operate in the foreground process. For nonoverlayed load modules, FRUN simply relocates the text and loads it into memory after first allocating space for the run-time stack at the beginning of the chain area. For the background process, the chain area begins at location 500 (octal) while, in a foreground process, it begins at the first location in the memory partition allocated to that process. In the case of overlayed load modules, a temporary file is created with a guaranteed unique name, and the text is relocated and written to this file. Subsequently, program requests to load an overlay reference this file in a manner similar to the nonrelocated load modules designed to operate in the background process.

FRUN is loaded near the end of the memory partition assigned to the process and is generally transparent to the calling program, normally the CLI. The program is also activated

through an RT-11 .CHAIN operation, with the name of the file to be stored in the chain area, terminated by a zero byte. The default device is DK, and the default file extension is REL. FRUN exits to the CLI (by an .EXIT programmed request) in case of error and otherwise to the loaded program. To the loaded program the entry from FRUN resembles a .CHAIN from the program that called FRUN; that is, the FRUN itself is transparent.

B.4 ARPANET VIRTUAL TERMINAL EMULATOR

User TELNET, the PRNET version, with minor modifications, is run as an application program in either a foreground or background process. It operates in single-character mode rather than line mode, so that line editing must be performed by the remote server process. This is normal for the ARPANET Tenex/TOPS-20 servers; however, for the BCN servers, line editing operations will probably be accomplished in the terminal process itself and are therefore not effective when using the TELNET program. Use of the existing SRI TELNET program is considered expedient until a more comprehensive TELNET program can be developed which can take advantage of BOS features such as multiple connections per process.

In the scenario most useful for internetworking experiments, TELNET is run in the foreground process. The program is started from the CLI by the RUN TELNET command. The controlling terminal must be in the single-character mode for TELNET to operate properly, so that the control-C interrupt does not work to abort the program to the CLI; however, the TELNET command @K causes exit to the CLI by the .EXIT programmed request. Since the CLI would probably overlay TELNET in this case, restart is not possible; thus, the TCP connection should be closed first.

B.5 RT-11 TELNET SERVER

The RT-11 TELNET server accesses the RT-11 emulator via a TCP-4 connection and a TELNET user program, such as the PRNET version which runs in the terminal interface unit. The present server implementation accesses all RT-11 emulator functions, including RT-11 compilers, utilities, and user programs such as FTP and TELNET.

The server requires about 400 words (including the TCB) and is designed to be dynamically loaded (for the TCB) near the end of the available storage area in either the foreground or background process. It is loaded and activated by the RUN command similar to standard RT-11 programs. Once activated, the process will accept and execute commands from either the TCP-4 connection or a DCN terminal process and is in general transparent to either source. However, since there is presently no way to initiate an asynchronous interrupt via the TCP-4 connection, it is not possible to interrupt a program via the usual control-C mechanism. Regardless, an asynchronous interrupt can be generated by closing and then reopening the connection using the appropriate remote TELNET user commands.

There are no explicit server commands other than those implied by opening and closing the connection. Once activated, the process listens on the local internet process address, port 27 (octal). When a connection request is received, the server saves the source address and henceforth accepts messages only from that source. Upon disconnect, the server resumes the original listening mode. No explicit login or logout sequence is expected or allowed.

For local logging, state changes during the connection open and close sequences cause the transmission of appropriate messages to a designated terminal process. Normally, this process is represented by the local operator's console, which can be

configured to send the messages to a local file or remote process, such as a TELNET user. Other features have not yet been implemented.

B.6 FILE TRANSFER PROTOCOL

An internet version of the ARPANET transfer protocol (FTP) has been developed which is designed to facilitate the transfer of files between DCN hostels and hosts on other networks. The server operates using TCP-4 as the transport mechanism and is capable of simultaneous, batched transmission of files in both directions. As in previous versions of the FTP operating on the ARPANET, this server can be controlled directly by a terminal or indirectly by a user program using a simple protocol. In either case, the control connection can be maintained within the local net using DCN messages or remotely using a TCP-4 connection and TELNET.

FTP is run as a user program in the background process and controlled by messages, with the actions taken indicated in replies to these messages. As in the ARPANET FTP, commands take the form of 4-character alphabetic names followed by an argument list separated by spaces. Replies are coded according to the new FTP format (RFC 640) when it is consistent with the enhanced capabilities provided. In all cases, the reply code consists of three digits, indicating the status of the action requested, the portion of the system affected, and additional detail, respectively. In some replies, the connection or file name, as appropriate, is included.

In general, reply codes beginning with the digit 1 indicate the start of a requested action and that commands may be given to initiate other unrelated actions. The action requested

is complete when a reply is generated beginning with the digit 2. Action is not initiated because of an error condition if a reply code is generated beginning with 4 or 5. This may occur even after a reply code beginning with 1 has been generated. The FTP makes no other distinctions in the reply codes; however, it does comply with the login and transfer parameter protocols and defaults specified in RFC 542 and RFC 640.

Data connections are established by TCP calls explicitly in response to the FTP CONN command or implicitly in response to a transfer request command such as RETR. In the latter case, the name of the remote internet FTP server is specified prior to the transfer request by a SOCK command. One of the two FTP servers at the end of the data connection is ordinarily set in the listening mode (null argument to the CONN command) before the other end tries to open the connection.

After the connection is established, it remains open for all transactions until it is closed explicitly by the DSCN or CONN commands or by logging off with the QUIT command. If the connection breaks due to net trouble, all transfers are aborted, but partially recorded information is not lost. In this case, the CLOS command can be used to close and preserve the contents of partially recorded files. Data transfers are initiated by STOR, RETR, LIST, and NLST commands in the same fashion as ARPANET FTP, with the added feature that a STOR operation can be active at the same time either a RETR, LIST, or NLST command is active; i.e., the connection is full-duplex. To designate the operation to which the particular reply applies, the 3-digit reply code is followed by either the text string STOR or RETR as appropriate.

In the absence of errors, a RETR operation terminates at the end-of-file and causes the TCP to set end-of-letter (EOL) in the last packet. The STOR operation terminates when the record corresponding to the last block preceeding EOL is written. To

allow multiple RETR operations or one host to append data in a single file on another such as may be appropriate for mail functions, the file written by a STOR operation is not closed following termination of that operation. Another STOR operation (with a null argument) initiated after termination of a previous STOR will cause data to be appended immediately following the last byte recorded. A CLOS or a STOR command with a non-null argument closes the current STOR file and, in the latter case, opens a new one. The file is also closed as a part of the BYE command processing.

The following lists the currently implemented FTP commands. Only the first three letters of the command name are significant. File names are structured according to the RT-11 connections in the form

device:file.ext[size]

where the device name (device) is two or three characters, the file name (file) consists of one to six RAD50 characters (in ASCII), the extension (ext) consists of one to three RAD50 characters, and size is the maximum file size in units of 256-word (512-byte) blocks. The size specification is optional and, in any case, applies only to STOR files. When omitted, the size initially allocated to the file is determined by the standard RT-11 rule: half the size of the largest contiguous free area on the device. Again, corresponding to the RT-11 rules, the device name defaults to DK and extension defaults to null.

HELP

Argument:	None.
Reply codes:	214 (last line), 114 (all other lines).
Function:	Prints list of FTP commands

NOOP

Arguments:	None.
Reply codes:	200
Function:	No-operation.

TYPE arg1 arg2

Arguments:	arg1 = A (ASCII), arg2 = N (nonprint).
Reply codes:	200, 501
Function:	Syntax check only (for compatibility).

STRU arg

Arguments:	arg = F (file)
Reply codes:	200, 501.
Function:	Syntax check only (for compatibility)

MODE arg

Arguments:	arg = S (stream)
Reply codes:	200, 501
Function:	Syntax check only (for compatibility)

BYTE arg

Arguments:	arg = 8 (bits/byte).
Reply codes:	200, 501.
Function:	Syntax check only (for compatibility).

STAT

Arguments:	None.
Reply codes:	211 (last line), 111 (all other lines)
Function:	Prints current status, including connection flags and name, STOR file information, and RETR file information.

USER arg

Arguments:	arg = user name (ignored).
Reply codes:	331.
Function:	Syntax check only (for compatibility).

PASS arg

Arguments:	arg = password (ignored).
Reply codes:	230.
Function:	Syntax check only (for compatibility).

BYE

Arguments:	None.
Reply codes:	221 (Also see DISC, CLOS, QUIT).
Function:	Equivalent to the command sequence DISC, CLOS, QUIT.

Notes: This command can be delayed pending TCP operations and device latencies.

QUIT

Arguments: None.
Reply codes: 221
Function: Exit to DCN/RT-11 monitor or TELNET server. This does not abort the file transfers or close the connections or files.

SOCK arg

Arguments: arg = host name (default listening).
Reply codes: 200, 501.
Function: Specify host address to be used for subsequent data connections.
Notes: If the host name is missing the connection is opened for listening. If the host name is invalid, reply code 501 is returned.

CONN arg

Arguments: arg - host name (default listening).
Reply codes: 200, 501 (See also SOCK, DISC).
Function: Equivalent to the command sequence SOCK DISC, followed by an attempt to open a TCP data connection to the address specified.
Notes: In the implied command sequence, if the SOCK fails, abort the sequence. If the SOCK succeeds, close the existing data connection (if necessary) with the implied DISC command,

which may be delayed pending TDP operations. The local TCP then tries to open the connection. When the connection is established, code 200 is returned. If the connection request is not satisfied in approximately 15 seconds, the request is aborted and code 425 is returned.

DISC

Arguments:	None.
Reply codes:	200 (See also ABOR command).
Function:	Equivalent to the ABOR command (no arguments) followed by an attempt to close the data connection.
Notes:	The implied ABOR command aborts all file transfers in progress and may cause code 426 messages to be returned. If a data connection was established, it is closed. When the closing handshake is complete, code 200 is returned. This command can be delayed pending TCP operations.

STOR arg

Arguments:	arg = filename (default indicates use filename specified in previous STOR).
Reply codes:	150, 250, 550, 553 (See also CLOS, CONN, ABOR).
Notes:	If a STOR operation is already in progress, code 550 is returned. If the file name is not properly structured, or it requests more storage than available, code 553 is returned.

This code is also returned if the file name is defaulted and was not specified previously in a STOR command. If the file is correctly specified (not defaulted), the file previously specified (if any) is closed by an implied CLOS command. Then the new file is opened on the specified device (default DK), and an attempt is made to open the data connection as per an implied CONN command (if not already open). This may be delayed pending TCP operations. If this is successful, code 150 is returned and the data are written to the file as they arrive from the net. When data transfer is complete, as indicated by an EOL from the net, code 250 is returned. If an error occurs during writing (for example, because of a hardware fault or exceeding the available device storage) code 451 is returned. If transfer is aborted due to a broken data connection or upon the request of an explicit or implicit ABOR commands, code 426 is returned, and the code 250 message will not be returned.

CLOS

Arguments:	None.
Reply codes:	200
Function:	Close the file opened by a STOR command.
Notes:	This command may be delayed pending device latencies.

RETR arg

Arguments:	arg = file name (see above).
Reply codes:	150, 250, 550, 553 (see also CONN, ABOR).
Notes:	See notes under STOR command. If the file is not found on the specified device (default DK), code 450 is returned. After transfer has begun, the corresponding TCP and remote FTP server must accept the data within about 15 s, or the TCP will break the connection and code 425 will be returned. If an error occurs during reading due to device or media problems, code 451 is returned. If data are refused by the TCP due to an error internal to the TCP or network, code 425 is returned.

LIST arg

Arguments:	arg = device name (default DK).
Reply codes:	(see RETR command)
Function:	Transmit a copy of the device directory on a specified device over data connection.
Notes:	See notes under RETR command.

NLST arg

Argument:	arg = device name (default DK).
Reply codes:	(See RETR command)
Function:	Transmit a copy of the device directory on a specified device over TELNET connection.

Notes: See the notes under the RETR command. This command does not require the data connection to be opened, so no return codes are associated with this operation, unless the data connection is already open and happens to close during the NLST operation.

ABOR arg1 arg2

Arguments: arg1, arg2 = S (STOR) or R (RETR) or default (both).

Reply codes: 200, 426.

Notes: If the specified data transfer(s) is in progress, one or more code 426 messages are returned and the transfer(s) is aborted. In any case, after the code 426 messages, code 200 is returned.

APPENDIX C. DESCRIPTION OF TIMING SIMULATION PROGRAM

The attached program listings describe the time-domain simulation program for clock tracking between two PSAT in the wide-band network. Figure C-1 shows the implemented simulation model. The main program in Figure C-2 accepts input error magnitudes and then "runs" the model for "MAX" frames. For a frame time of 20 ms, a run of 10^4 frames would simulate 200 s of real time. The basic variables are updated every DELT seconds where DELT = 1 ms.

The operation of two stations is simulated; one is designated the master (M) and the other the slave (S). The master station transmits a reference burst once per frame. This burst may experience slight "jitter" with respect to some hypothetical "true" time because of clock jitter and offsets and small random delay variations in the earth station interface (ESI). The program deals with these small departures from true or absolutely perfect timing. They are measured as radians of the symbol clock (i.e., one clock cycle at 1.544 Msymbols/s = 2π radians). After transmission, the errors in timing epochs experience delays and path-length variations due to satellite motion. These delays are introduced by the routine DELAY( ) in Figure C-3, which acts as a 128-ms delay for the up- and down-link transmissions. The five transmissions are kept separate: $(M)_{up}$, $(S)_{UP}$, $(M \rightarrow M)_{DN}$, $(M \rightarrow S)_{DN}$, and $(S \rightarrow S)_{DN}$.

The routine DOPP( ) in Figure C-4 adds the Doppler shift due to path-length motion. The subroutine JITTER( ) adds perturbations in the local-time clocks at the master and slave stations. These oscillator jitters are composed of three components: white phase noise, $1/f^2$ phase noise, and a fixed offset in frequency.

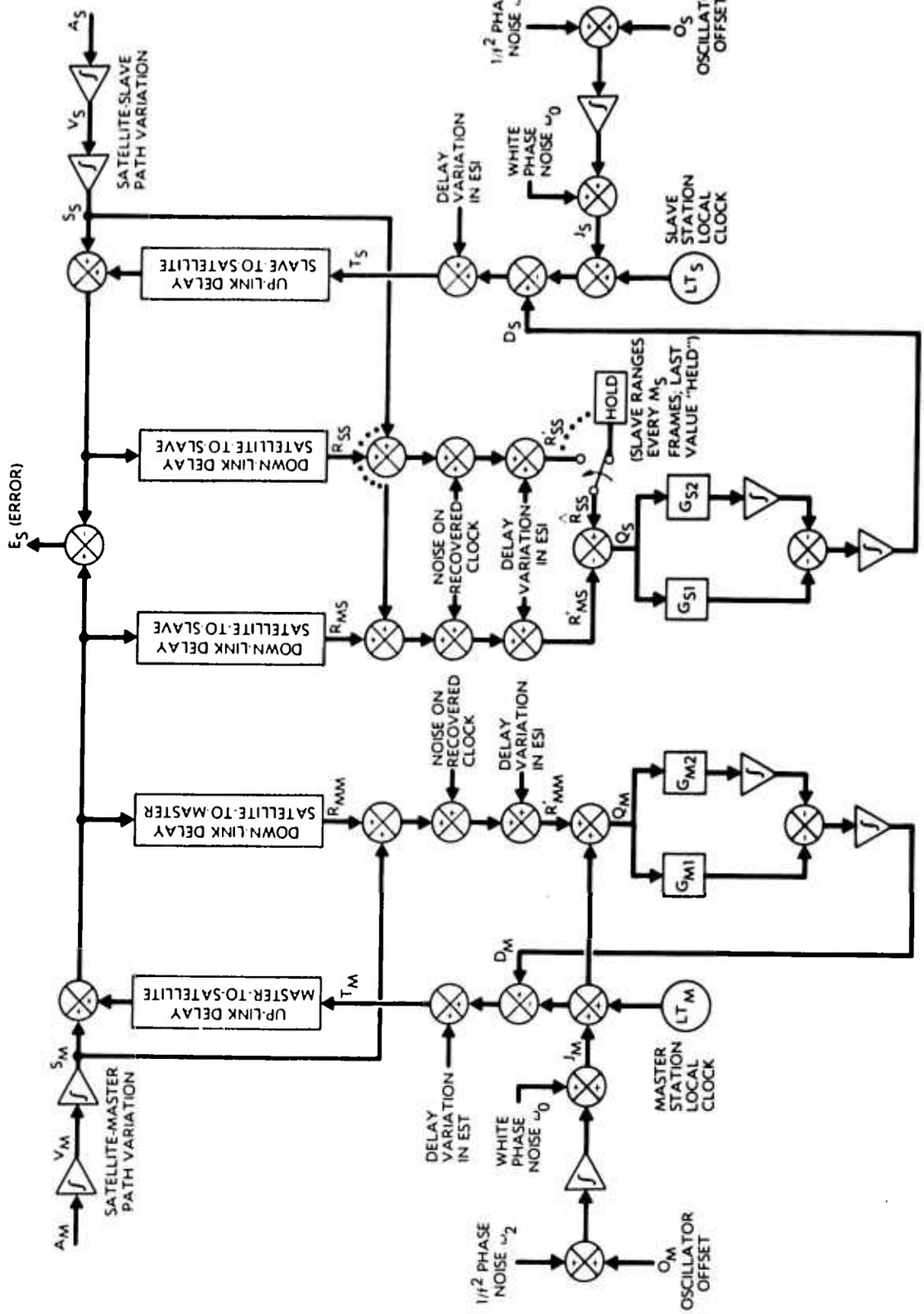


Figure C-1. Timing Error Model

```
C DECEMBER 1979
C   DECK # 2
C PROGRAM TO SIMULATE TIMING ERRORS BETWEEN TWO STATIONS IN THE EISN
C SYSTEM.  FACTORS INCLUDE:
C      1) SATELLITE MOTION
C      2) OSCILLATOR SHORT-TERM STABILITY (PHASE NOISE)
C      3) DELAY VARIATIONS IN THE ESI OR THE PSAT
C      4) NOISE THAT CAUSES ERRORS IN CLOCK RECOVERY
C      5) THE ALGORITHM USED TO COMPUTE GLOBAL TIME
C      6) QUANTIZING ERRORS
C A BASIC SAMPLING INTERVAL OF ONE MILLISECOND IS USED IN THE
C SIMULATION.  MAXIMUM TIME DELAYS EARTH-TO-SATELLITE ARE 128 MILLISEC.
C      INDEX FOR LINKS; 1=M TO M,2= M TO S, 3= S TO S
C THE CONVERSION FACTORS ARE: RTOT= RADIANS -TO-SECONDS; DTOR=METERS-TO-
C  RADIANS
      DIMENSION DSVALS(100),DMVALS(100),ESVALT(100),TIMES(100)
      COMMON/COM1/IENT1,IENT2,ISEED
      NPTS=10
      NPTS=100
      STEP=1.
      STEP=0.
C  MASTER AND SLAVE PATH ACCELERATION (METERS/SECOND**2)
      DATA AM/0./,AS/0./
C  MASTER AND SLAVE PATH VELOCITIES (METERS/SECOND)
      DATA VM/0./,VS/0./
C SOURCES OF ERROR IN THE SYSTEM ARE DETERMINED BY THE FOLLOWING
C PARAMETERS:  SIGMTO-STD DEV OF WHITE PHASE JITTER MASTER L.T. CLOCK
C      SIGSTO- STD DEV OF PHASE JITTER, SLAVE L.T. CLOCK
C      VALUES ARE IN RADIANS
      DATA SIGMTO/0.00707/,SIGSTO/0.00707/
C      ESITM-TIME ERROR IN MASTER ESI ON TX
C      ESITS-TIME ERROR IN SLAVE ESI ON TX
C      RANGE OF XMT ESI DELAYS (# OF SYMBOLS)
      DATA ERTXX/1.0/
C      SRXM- RMS ERROR IN CLOCK RECOVERY AT MASTER
C      SRXS- RMS ERROR IN CLOCK RECOVERY AT SLAVE
C      ERRORS ARE IN DEGREES
      DATA SRXM/10./,SRXS/10./
C      ESIRMM-TIME ERROR IN RX ESI M-TO-M
C      ESIRMS- TIME ERROR IN RX ESI M-TO-S
C      ESIRSS- TIME ERROR IN RX ESI S-TO-S
C      RANGE OF RCVE ESI DELAYS (# OF SYMBOLS)
      DATA ERRXX/1.0/
C      VARM-ST DEV (F)**-2 LT CLOCK NOISE,MASTER
C      VARS-ST DEV (F)**-2 LT CLOCK NOISE, SLAVE
C      RADIANS
      DATA VARM/0.003/, VARS/0.003/
C OSCILLATOR "OFFSETS" IN PARTS PER MILLION(PPM)
```

Figure C-2. Program to Simulate Timing Errors (1 of 4)

```
      DATA OOM/2.0/, OOS/-2.0/
      GM1=2.
      GM2=1.
      GS1=1.
      GS2=1./4.
C  SAMPLES PER FRAME
      IFRAME=20
C SAMPLING INTERVAL (SECONDS)
      DELT=1.E-3
      TFRAME=DELT*IFRAME
C #OF FRAMES TO RUN
      MAX=1000
      MAX=300
      MAX=10000
C  MASTER TO SLAVE RATIO OF TIMING UPDATES
       MSRAT=20
      MSRAT=64
      MSRAT=32
      MSRAT=16
       ISLAVE=MSRAT
      RXSS=0.
      RXMM=0.
      RXMS=0.0
C     SYMBOL RATE  (SYMBOLS/SECOND)
      RS=1.544E+6
      PI=3.1415926
C  CONVERT TO RADIANS
      ERTX=2.*PI*ERTXX
      ERRX=2.*PI*ERRXX
      SRXM=SRXM/57.29578
      SRXS=SRXS/57.29578
      VLITE=3.E+8
      RTOT=1./(2.*PI*RS)
      DTOR=(2.*PI*RS)/VLITE
      DTOT=1./3.E+8
      OFFM=OOM*RS*2.*PI*DELT/1.E+6
      OFFS=OOS*RS*2.*PI*DELT/1.E+6
      DM=0.
      DDOTM=0.
      DS=0.
      DDOTS=0.0
      IS=0
      ISAMP=0
      NFRAME=1
      CALL STAT(1,0.)
C##########################################################
C##########################################################
C##########################################################
```

Figure C-2. Program to Simulate Timing Errors (2 of 4)

```
C BEGIN THE SIMULATION AT A SAMPLE COUNT OF 1 AND A FRAME COUNT OF 1
C  UPDATE SAMPLE COUNT AND FRAME COUNT
  1    IS=IS+1
       ISAMP=ISAMP+1
       IF(IS .GT. IFRAME) GO TO 3
       GO TO 4
     3 IS =1
       NFRAME=NFRAME+1
        ISLAVE=ISLAVE+1
     4 CONTINUE
       CALL DOPP(AM,AS,VM,VS,DELT,SM,SS)
       DOPM=SM*DTOR
       DOPS=SS*DTOR
       CALL JITTER(VARM,VARS,OFFM,OFFS,PHERM,PHERS )
       IF(IS  .EQ.  1) GO TO 10
       GO TO 11
    10 CONTINUE
C IF FIRST SAMPLE IN FRAME,MASTER STATION TRANSMITS
C WHEN ISV1, TIME FOR BOTH MASTER AND SLAVE TO TRANSMIT
       ESITM=(GGUBF(ISEED)-0.5)*ERTX
       ESITS=(GGUBF(ISEED)-0.5)*ERTX
       TM=OM-PHERM-GGNOF(ISEED)*SIGMTO+ESITM+STEP
       TS=OS-PHERS-GGNOF(ISEED)*SIGSTO+ESITS
       CALL DELAY(TM,TS,DOPM,DOPS,TOMM,TOMS,TOSS,ES  )
       GO TO 14
    11 CONTINUE
       IF(IS .EQ.  17) GO TO 12
       GO TO 13
   12  CONTINUE
C##################################################################
C CORRECT THE  'D'S' BASED ON THE RECEIVED BURSTS:  THE STEPS ARE:
C      1) NOT TIME TO TRANSMIT SO INPUTS TO DELAY ARE 0
       TM=0.
       TS=0.
       CALL DELAY(TM,TS,DOPM,DOPS,TOMM,TOMS,TOSS,ES  )
C      2) CHECK TO SEE IF ROUND-TRIP TIME HAS ELAPSED; IF NOT,DO NOTHING
       IF(ISAMP  .GT.  256) GO TO 16
       GO TO 17
C      3) ADD DOPPLER EXPERIENCED ON DOWN LINKS PLUS RX NOISE
    16 CONTINUE
       ESIRMM=(GGUBF(ISEED)-0.5)*ERRX
       ESIRMS=(GGUBF(ISEED)-0.5)*ERRX
       RXMM=TOMM+SM*DTOR+GGNOF(ISEED)*SRXM+ESIRMM
       RXMS=TOMS+SS*DTOR+GGNOF(ISEED)*SRXS+ESIRMS
C      4) NOW CORRECT THE  'D' VALUES (CHANGE THIS PART FOR OTHER
C       CORRECTION STRATEGIES )
C THE ERROR SIGNALS ARE OM AND OS
       OM=RXMM+PHERM+GGNOF(ISEED)*SIGMTO
```

Figure C-2. Program to Simulate Timing Errors (3 of 4)

```
      QS=RXSS-RXMS
      DM=DM-(DDOTM+GM1*QM)*TFRAME
      DS=DS-(DDOTS+GS1*QS)*TFRAME
      DDOTM=DDOTM+GM2*QM*TFRAME
      DDOTS=DDOTS+GS2*QS*TFRAME
       IF(ISLAVE.GT.MSRAT) GO TO 100
       GO TO 17
  100 ISLAVE=0
      ESIRSS=(GGUBF(ISEED)-0.5)*ERRX
      RXSS=TOSS+SS*DTOR+GGNOF(ISEED)*SRXS+ESIRSS
   17 CONTINUE
C#####################################################################
      GO TO 14
   13 CONTINUE
      TM=0.
      TS=0.
      CALL DELAY(TM,TS,DOPM,DOPS,TOMM,TOMS,TOSS,ES  )
   14 CONTINUE
      IF(IS  .EQ. 9) GO TO 18
      GO TO 19
   18 CONTINUE
C     WRITE(6,55) IS,NFRAME,QM,DDOTM,DM,QS,DDOTS,DS,ES
C  55 FORMAT(1X,2(I6,3X ),7(F9.4,2X))
      ERR=ES*RTOT
      CALL STAT(2,ERR)
C     ESVALT(NFRAME)=ES
   19 CONTINUE
      IF(NFRAME.EQ.MAX) GO TO 15
      GO TO 1
C#####################################################################
C#####################################################################
C#####################################################################
   15 CONTINUE
      CALL STAT(3,0.)
      END
```

Figure C-2. Program to Simulate Timing Errors (4 of 4)

```
      SUBROUTINE DOPP(AM,AS,VM,VS,DELT,SM,SS)
      COMMON/COM1/IENT1,IENT2,ISEED
C COMPUTES RANGE VARIATION DUE TO SATELLITE MOTION:
C     AM,AS; MASTER & SLAVE PATH ACCELERATION (M/S**2)
C     VM,VS; MASTER & SLAVE PATH VELOCITY (M/S)
C     SM,SS; MASTER & SLAVE PATH CHANGE (M)
C     DELT; SAMPLING INTERVAL
      IF(IENT1 .EQ. 1) GO TO 1
      VVM=VM
      SM=0.
      VVS=VS
      SS=0.
    1 CONTINUE
      VVM=VVM+AM*DELT
      SM=SM+VVM*DELT
      VVS=VVS+AS*DELT
      SS=SS+VVS*DELT
      IENT1=1
      RETURN
      END
```

```
      SUBROUTINE JITTER(VARM,VARS,OFFM,OFFS,PHERM,PHERS )
      COMMON/COM1/IENT1,IENT2,ISEED
      IF(IENT2 .EQ. 1) GO TO 2
      PHERM=0.
      PHERS=0.
    2 CONTINUE
      PHERM=PHERM+GGNOF(ISEED)*VARM+OFFM
      PHERS=PHERS+GGNOF(ISEED)*VARS+OFFS
      IENT2=1
      RETURN
      END
```

```
      BLOCK DATA
      COMMON/COM1/IENT1,IENT2,ISEED
      DATA IENT1/0/
      DATA IENT2/0/
      DATA ISEED/1979/
      END
```

Figure C-3. Subroutine Dopp

```
      SUBROUTINE DELAY(TINM,TINS,DOPM,DOPS,TOMM,TOMS,TOSS,ES     )
C     SIMULATES MULTIPLE DELAY PATHS, ONE PER UP AND DOWN LINK
C     INDEX OF DUP(); 1=MASTER,2=SLAVE,3-4 NOT USED
C     INDEX OF DDN() ; 1=MM   ,2=MS ,3=SS, 4= NOT USED
C     INDEX OF POINTERS: 1= M UP; 2= S UP; 3-4=NOT USED; 5=MM DOWN;
C                        6=MS DN; 7=SS DOWN
      DIMENSION DUP(4,129),DDN(4,129),JIN(8),JOT(8),JMX(8)
      DATA DUP/516*0./,DDN/516*0./,JIN/8*1/,JOT/8*2/,JMX/8*129/
      DUP(1,JIN(1))=TINM
      DUP(2,JIN(2))=TINS
      DDN(1,JIN(5))=DUP(1,JOT(1))+DOPM
      DDN(2,JIN(6))=DUP(1,JOT(1))+DOPM
      DDN(3,JIN(7))=DUP(2,JOT(2))+DOPS
      ES=DDN(2,JIN(6))-DDN(3,JIN(7))
      TOMM  =DDN(1,JOT(5))
      TOMS  =DDN(2,JOT(6))
      TOSS  =DDN(3,JOT(7))
      DO 1 L=1,8
      JIN(L)=JIN(L)+1
      IF(JIN(L) .GT. JMX(L)) JIN(L) =1
      JOT(L)=JOT(L)+1
      IF(JOT(L) .GT. JMX(L)) JOT(L)=1
    1 CONTINUE
      RETURN
      END
```

Figure C-4. Subroutine Dopp

The slave station tracks the received master reference burst, and every MSRAT frames, transmits its own ranging burst to measure round-trip time (RTT). Based on the reference epoch and RTT, the slave station attempts to correct its timing epoch so that its up-link transmissions arrive at the satellite at the same time as those of the master. This is an artifice used in the simulation; it is assumed that the master and slave station are attempting to transmit bursts that arrive at the satellite simultaneously. The error in arrival time (ES) measures their inability to do this precisely and is the main output of the simulation.

The subroutine STAT( ) shown in Figure C-5 processes the errors once per frame and determines the mean, variance, and cumulative distribution. The first "NULL" of these errors (typically 500 frames) is discarded to allow the model to reach a steady-state condition.

The model shown assumes that second-order tracking is used at both the master and slave stations. With this type of tracking, the master and slave clocks can, in effect, remain phase locked even with rather large frequency offsets in the LT clocks, and with path length velocity and acceleration components. Other tracking algorithms could be simulated by changing a few statements in the program as noted.

```
      SUBROUTINE STAT(IOPT,ERRX)
C  COMPUTES MEAN ERROR AND R.M.S. TIMING ERRORS IN SYMBOLS
C     ALSO COMPILES THE CUMULATIVE DISTRIBUTION OF TIMING ERRORS
      DIMENSION IHIST(100),CHIST(100)
      DIMENSION XVAL(100)
      NCELL=100
      NULL=500
      RS=1.544E+6
      CELL=0.2
      GO TO (1,2,3),IOPT
    1 SUM=0.
      SUMSQ=0.
      ICOUNT=0
      SPAN=NCELL*CELL
      XBIG=SPAN/2.
      XLIT=-XBIG
      DO 6 I=1,NCELL
      IHIST(I)=0
    6 CHIST(I)=0.
      IBIG=0
      ILIT=0
      GO TO 4
    2 ERROR=ERRX*RS
      ICOUNT=ICOUNT+1
      IF(ICOUNT .LT. NULL) GO TO 8
      SUM=SUM+ERROR
      SUMSQ=SUMSQ+ERROR*ERROR
      IF(ERROR .GE. XBIG) GO TO 7
      IF(ERROR .LE. XLIT) GO TO 9
      XYZ=ERROR/CELL+NCELL/2
      INDEX=XYZ+0.9999999
      IHIST(INDEX)=IHIST(INDEX)+1
      GO TO 8
    7 IBIG=IBIG+1
      GO TO 8
    9 ILIT=ILIT+1
    8 CONTINUE
      GO TO 4
    3 XXX=ICOUNT
      XXX=XXX-NULL
      XMEAN=SUM/XXX
      Y=SUMSQ/XXX
      SIG=SQRT(Y-XMEAN*XMEAN)
      WRITE(6,5) XMEAN,SIG
    5 FORMAT(1X,'MEAN ERROR=',F8.3,2X,'STD.DEV.=',F8.3,2X)
      CSUM=0.
      CSUM=CSUM+IBIG
      DO 10 K=1,NCELL
```

Figure C-5. Subroutine Stat (1 of 2)

```
      KK=NCELL+1-K
      ERR=CELL*(KK-50.-0.5)
      FR=CSUM/(ICOUNT-NULL)
      WRITE(6,11) ERR,FR
   11 FORMAT(1X,'ERROR', F8.3,2X,'FRACTION EXCEEDED=',F8.5,2X  )
      CSUM=CSUM+IHIST(KK)
   10 CONTINUE
C  PLOT THE HISTOGRAM OF ERRORS
      XINTV=XLIT-CELL
      XCOUNT=ICOUNT-NULL
      DO 100 I=1,NCELL
      CHIST(I)=IHIST(I)/XCOUNT
      XINTV=XINTV+CELL
      XVAL(I)=XINTV
  100 CONTINUE
C     CALL GRAPH(NCELL,XVAL,CHIST,3,7,10.,8.,0.,0.,0.,0.,
C    1          'MICROSECONDS;','FREQ.;','HISTOGRAM;','OF ERRORS;')
    4 RETURN
      END
```

Figure C-5. Subroutine Stat (2 of 2)