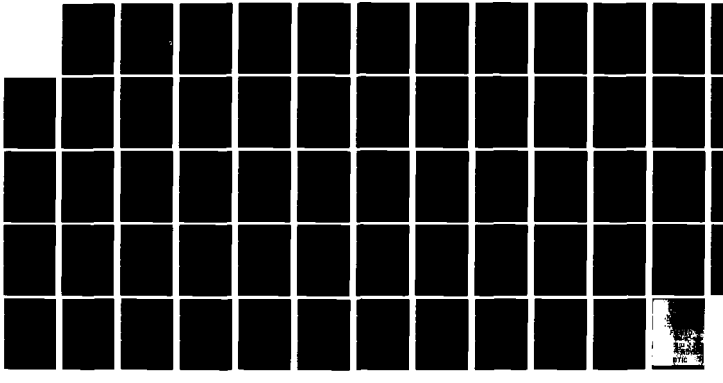
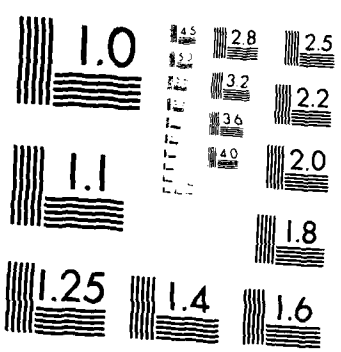
AD-A129 828 SATNET (ATLANTIC PACKET SATELLITE EXPERIMENT) DEVELOPMENT AND OPERATION P.. (U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA J F HAVERTY MAY 83 BBN-5345 1/1

UNCLASSIFIED MDA903-80-C-0353 F/G 17/2 NL





Bolt Beranek and Newman Inc.

Report No. 5345

Combined Quarterly Technical Report No. 29

SATNET Development and Operation
Pluribus Satellite IMP Development
Internet Operations and Maintenance
Mobile Access Terminal Network

May 1983

Prepared for:
Defense Advanced Research Projects Agency

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A124828	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Combined Quarterly Technical Report No. 29	5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical 2/1/83 to 4/30/83
	6. PERFORMING ORG. REPORT NUMBER 5345
7. AUTHOR(s) J. F. Haverty	8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0353 N00039-81-C-0408
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order No. 3214
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209	12. REPORT DATE May 1983
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DSSW Room 1D Pentagon Washington, DC 20310 NAVELEX Washington, DC 20360	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE/DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer networks, packets, packet broadcast, satellite communication, gateways, UNIX, Pluribus Satellite IMP, shipboard communications, ARPANET, Internet

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Quarterly Technical Report describes work on the development of and experimentation with packet broadcast by satellite; on development of Pluribus Satellite IMPs; on Internetwork monitoring; and on shipboard satellite communications.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 5345

COMBINED QUARTERLY TECHNICAL REPORT NO. 29

SATNET DEVELOPMENT AND OPERATION
PLURIBUS SATELLITE IMP DEVELOPMENT
INTERNET OPERATIONS AND MAINTENANCE
MOBILE ACCESS TERMINAL NETWORK

May 1983

This research was supported by the Defense Advanced Research Projects Agency under the following contracts:

MDA903-80-C-0353, ARPA Order No. 3214
N00039-81-C-0408

Submitted to:

Director
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Attention: Program Management

Table of Contents

TABLES

1 INTRODUCTION

This Quarterly Technical Report is the current edition in a series of reports which describe the work being performed at BBN in fulfillment of several ARPA work statements. This QTR covers work on several ARPA-sponsored projects including (1) development and operation of the SATNET satellite network; (2) development of the Pluribus Satellite IMP; (3) Internet Operations, Maintenance, and Development; and (4) development of the Mobile Access Terminal Network. This work is described in this single Quarterly Technical Report with the permission of the Defense Advanced Research Projects Agency. Some of this work is a continuation of efforts previously reported on under contracts DAHC15-69-C-0179, F08606-73-C-0027, F08606-75-C-0032, MDA903-76-C-0214, MDA903-76-C-0252, N00039-79-C-0386, and N00039-78-C-0405.

2 SATNET DEVELOPMENT AND OPERATION

The major emphases placed on our participation in the Atlantic Packet Satellite Experiment (SATNET) were the Satellite IMP software maintenance operations and the overall SATNET hardware maintenance operations, which are described in the following sections; some of our other activities are described immediately below.

We ordered and took delivery of two standard BBNCC C/30 packet switch processors, designated to replace the Honeywell 316 Satellite IMPs at Goonhilly and Tanum. Both machines were placed in the 220 volt testbed at BBN to undergo a thorough burn-in session. In the first week, standard ARPANET test programs were run to test processor, memory, and standard interfaces. In the second week, the I/O interface daughter boards for the PSP terminal were attached, and the Satellite IMP software was run. While in the testbed, the Goonhilly C/30 halted repeatedly during one series of tests; although a C/30 I/O board malfunction was initially suspected, subsequent tests indicated that the problem was probably due to an error in seating the daughter boards on the I/O board.

We transferred the SATNET NU monitoring programs to the BBN-NET host NOC2 C/70 UNIX computer to serve as a backup to the primary SATNET monitoring host, the BBN-NET host INOC C/70 UNIX

computer. NU is also used for monitoring the ARPANET, the WIDEBAND satellite net, gateways, and the BBN-NET, among others; commonality of network monitoring programs facilitates software maintenance.

Work on incorporating the Native Mode Firmware System (NMFS) in Satellite IMPs continued (in NMFS, the emulated machine is altered to create one more suited to communications applications, that is, having greater throughput capacity and reduced latency).

2.1 Software Maintenance Operations

The NU ltbox program was expanded to provide matrix displays of the average frequency offsets and the average receiver AGC values, where the raw data are generated by the PSP terminal for every received packet and transferred to the Satellite IMP as part of the Testing and Monitoring (T&M) data. This information is now more readily absorbed for tracking changes in the system. The T&M data handler in the Satellite IMP macrocode was modified to accumulate the raw frequency offset and receiver AGC values before sending to NU. Originally we implemented in the Satellite IMP a seven-bit divide operation on Eb and No values before summing to provide channel signal-to-noise information; however, COMSAT determined in consultation with Linkabit that the values generated by the PSP terminal were insufficiently linear in the

range of interest to adequately reflect network behavior. Accordingly, we disabled the Eb and No processing to reduce the computational load on the processor. Typical matrix displays of the frequency offset and the receiver AGC are shown below.

At: \ From:	Etam	Goonh	Tanum	Raist	Clark
Etam	67	-761	180	-57	0
Goonh	-120	709	-230	6	0
Tanum	113	944	1	236	0
Raist	-400	433	-507	-272	0
Clark	0	0	0	0	0

SATNET Frequency Matrix (Hz)

At: \ From:	Etam	Goonh	Tanum	Raist	Clark
Etam	42	59	63	33	0
Goonh	74	94	99	58	0
Tanum	34	49	55	27	0
Raist	90	104	120	78	0
Clark	0	0	0	0	0

SATNET AGC Matrix

At the time these matrices were generated, the Clarksburg Satellite IMP was not connected to a PSP terminal, and consequently no T&M data were received.

We found and corrected several problems which appeared in the stream handling code when we stressed the system with the SATNET automatic stream facility for datagram traffic. On 7 February 1983, this facility was re-enabled, providing users with 1-satellite hop response times for short interactive TCP traffic.

2.2 Hardware Maintenance Operations

During the reporting interval, we helped to diagnose several hardware problems which appeared, and fixed those that were related to the Satellite IMPs. Mostly, the satellite channel provided good service, a large part of which is due to COMSAT's careful monitoring of the satellite channel.

One of the more frustrating problems was the week-long outage of the terrestrial circuit between the CSS gateway and the Etam Satellite IMP. Problem isolation and circuit repair consumed large amounts of time with WUI, TELCO, Etam, and BBN all involved. Finally, TELCO restored the circuit by undoing a data inversion on both sides of the circuit and by momentarily placing the Bell 303 modem at Etam in a local loopback mode. Fortunately, during this time the circuit between the DCEC gateway and the Etam Satellite IMP was operational to maintain SATNET connectivity to ARPANET.

The DCEC gateway crashed into its loader and refused to accept gateway software reloads from the NOC. Originally we suspected a PDP-11 memory problem, but eventually DCEC personnel determined that the ACC VDH-11C PDP-11 interface failed the diagnostics. Repair of the VDH interface required its return to ACC headquarters, a procedure which kept the terrestrial circuit non-operational for over five weeks.

During the interval when the DCEC VDH interface was being repaired, several problems occurred on the circuit between the CSS IMP and the Etam Satellite IMP, thus isolating the European internet due to the lack of a terrestrial access to Etam. First, after the circuit malfunctioned, WUI was called and discovered an open in the circuit. Second, work on the CSS site's air conditioning system required the CSS IMP to be powered off. Third, after the circuit malfunctioned again, WUI determined the source to be a TELCO problem. TELCO eventually found two problems, an open wire in the TELCO central office and a tip-ring inversion on one of the lines. The circuit was restored to service, after having been out for more than four days, one of which is attributable to TELCO repair personnel being denied access to CSS on a Sunday.

Power problems wreaked havoc with multiple SATNET sites on 21 February 1983. At Raisting, a primary power outage

necessitated reload of Satellite IMP memory; at Tanum, primary power cycled on and off at 10-second intervals; at Goonhilly, a major interruption occurred in the primary power servicing the SPADE equipment, the Satellite IMP, and the PSP terminal. One month later, British Telecom carried out urgent engineering work on the power supply systems at Goonhilly, resulting in a complete one-day shutdown of the Satellite IMP and the PSP terminal.

Raisting was found to be transmitting a shorter modem preamble than the other sites; COMSAT fixed the modem preamble by changing switch positions in the PSP terminal. Problems developed with the IF subsystem at Raisting when the unit was removed without powering off the PSP terminal. (Raisting is the only site that does not have access to a SPADE system and so is given extra IF circuitry to simulate the full SPADE terminal.) DFVLR temporarily swapped the old unit for a new unit at COMSAT to restore service.

When COMSAT and DFVLR were working with Raisting, they inadvertently left the Raisting IF subsystem's pilot switch on. Immediately thereafter, the following channel characteristics were observed.

- o No regular monitor reports from any of the European sites were being received at Etam. The European sites would periodically be declared "not heard," after which a poll-request packet would be sent by the monitoring host INOC. Most of the ensuing poll response packets would get through

the network, however, as would some of the trap reports from the European sites.

- o The monitor reports from Etam showed abnormally large numbers of packets received with errors in the hardware cyclic redundancy checksum (CRC) but not in the software header checksum.
- o Inhibiting Raisting Satellite IMP transmissions while leaving the pilot signal on halved the numbers of packets with CRC errors; nevertheless, the numbers were still abnormally large. Monitoring reports from the European sites were able to get through the network, however.
- o During this entire period, Etam heard all Hello packets from all sites correctly.

When the pilot signal was turned off, the error performance returned to normal. These observations are consistent with a syndrome in which short packets are able to traverse the satellite channel satisfactorily, while a substantial number of long packets fails -- a phenomenon we reported prevalent last fall. The implication is that the severe problems on the satellite channel last fall may have been due to a spurious pilot signal on the channel.

On several occasions, we noticed that the SATNET sites could not hear any Raisting Hello packets, although Raisting round-trip packets were definitely interfering with channel operations. We hypothesized that the Raisting receive channel malfunctioned. Not hearing any of its round-trip packets, the Satellite IMP failed to acquire frame synchronization but continued to transmit these packets every PODA frame. With no direct access to the

site, the Network Operations Center was unable to command the Raisting Satellite IMP to inhibit transmissions; hence, we had to telephone the earth station to request that the transmit cable be disconnected.

DFVLR later determined that the source of the problem is due to the PSP terminal being connected to only one of the two receive converters in the SPADE system at the Raisting earth station. The first converter is normally operational while the second remains on a hot stand-by basis. Switching between converters is done automatically whenever certain malfunctions, such as a power level drift of more than 3 dB, are detected in the first converter. Whenever this happens, the Raisting PSP terminal receive side becomes disconnected from the channel, while the transmit channel remains intact. DFVLR is examining ways of remedying the problem.

3 PLURIBUS SATELLITE IMP DEVELOPMENT

3.1 Introduction

BBN activities during the quarter concentrated on Wideband Satellite Network operations, preparation for the Wideband meeting, support of the Wideband Network Task Force, and BSAT software development.

3.2 Wideband Meeting and Task Force Activities

In preparation for the Wideband Meeting on March 16 and 17, BBN conducted extensive testing of the PSATs, ESIs and satellite channel at many combinations of data rate, coding level and modulation type. The results of this testing, presented at the meeting, indicated that there were some serious problems with the ESIs and the satellite channel. Data presented by Lincoln Laboratory and ISI also showed that more effort should be put into keeping the network up for a higher percentage of the time to provide reliable service to the network users. A task force, including Lincoln Laboratory, BBN, ISI and Linkabit, was formed to address these issues. As a first phase of the task force's plan, BBN repeated one of the previous channel tests in a very controlled manner at several sites, and the data gathered was delivered to Lincoln Laboratory and Linkabit for further study.

Based on an analysis of the data, Linkabit identified several problems in the ESI. Various fixes were proposed and tested at Linkabit. A Linkabit technician visited Lincoln and DCEC on April 14 and 15 to install these fixes in the ESIs and Burst Test Modems (BTM). Following his vist, the ESIs and BTMs at both sites were rendered inoperable. BBN worked with Linkabit engineers who visited Lincoln during the week of April 25 to further investigate the ESI's behavior.

It was found that the ESI did not properly process aggregated multi-rate coded bursts. It was also found that an ESI software bug limited a burst to 5 state transitions. The maximum allowed number of state transitions should be 16.

3.3 Network Operations, Maintenance, and Status

During February, the ISI earth station experienced problems with its high power amplifier (HPA). After several attempts to fix the problem, a spare 125 watt HPA was installed on February 22. The RF loopback relay at ISI failed during February and was repaired by Western Union on March 18. During the week of March 21, the equipment in the earth station shelter on the roof at ISI was moved to a location in the 12th floor computer room. The 125 watt HPA failed on March 28 and was replaced by a 75 watt spare unit. During April, the earth station's upconverter was found to

repeatedly drop out of lock. Several attempts were made to adjust it. Finally on April 21, Western Union was able to get it set correctly. Burst Test Modem (BTM) testing between ISI and Lincoln at the end of April showed a significant frequency offset for the ISI earth station. At the end of the quarter, Western Union was working on this problem.

Both the Lincoln and DCEC sites experienced outages during February, due to snow that had collected in the earth station antenna. The downconverter at DCEC failed on February 23 and was replaced by Western Union. The 125 watt HPA failed at DCEC on March 7 and was replaced with a spare 75 watt unit. On March 21, the Lincoln HPA failed and was replaced on March 29 with a repaired unit which had originally come from the RADC. This repaired unit experienced many intermittent problems throughout April.

The ISI ESI continued to have difficulties throughout the month of February. On March 3, a spare ESI Interface Codec and Control Unit (ICCU) was installed at ISI. It contained PROMs with incorrect default channel parameters and would not work properly with the other sites. On March 18, new PROMs were installed and ISI was able to communicate with the other sites.

As a part of the Task Force effort, a Linkabit technician visited Lincoln and DCEC on April 14 and 15 to install a series

of fixes in the ESIs and BTMs. The ESIs and BTMs at both sites were rendered inoperable following the technician's visit. Linkabit engineers visited Lincoln during the week of April 25 to install the fixes correctly and were able to get the Lincoln ESI and BTM working again. The DCEC ESI was sent back to Linkabit for repair and the DCEC BTM was removed from the site by Western Union.

BBN installed redundant Satellite Modem Interfaces (SMIs) and Super Sue Pollers in the PSATs at ISI on February 2 and at Lincoln on March 11. Both PSATs at ISI and Lincoln experienced minor problems with their cassette readers during the early part of February. The Lincoln PSAT developed a hardware problem on March 18. BBN field service traced the problem to a bus coupler between the P20 processor bus and the E000 I/O bus. On March 25, the ISI PSAT developed hardware problems. The problem was identified as an intermittent fault in an I/O bus power supply, which was temporarily swapped with a processor bus power supply to get the machine up and running. The faulty power supply was eventually replaced on April 18. The ISI PSAT cassette reader failed on April 26 and was replaced.

3.4 PSAT Software Development

A new version of the PSAT software was tested on the network during February. The new version provided for the configuration of the RADC PSAT's hosts. It also reduced the size of the FPODA control subframe from 9 to 4 slots. At this time control subframe slots are included only for the 5 sites that currently exist within the network. The leader PSAT uses the leader subframe for its control packets.

The new PSAT software included code to collect Test and Measurement (T&M) data from the ESI. New NU software was added to receive and display the T&M data. Several problems were encountered with the ESIs at the various sites when T&M collection was enabled. At SRI, no T&M packets were ever received from the ESI. T&M collection could be successfully turned on at ISI, Lincoln and DCEC. However, it would only run for at most 15 minutes before completely stopping. Linkabit found and corrected several bugs in the ESI processing of T&M data. The ordering of a couple of T&M words within a T&M burst was swapped to match the documentation, and the problem of T&M processing by the ESI occurring for only a short time after it was enabled by the PSAT was fixed. It will now run continuously.

Another new version of the PSAT code was tested at BBN and Lincoln Laboratory during April. This new version converts the

spare PSAT code pages into additional message buffers to allow the PSAT to handle greater levels of host traffic. The new version of the PSAT program has obsoleted the host version of the PSAT loader program. Cassettes with the new loader will be distributed to the other sites shortly to allow the new software to be tested there.

3.5 Summary of BSAT Software Development

During February, software was added to gracefully shut down the multiprocessor BSAT system. The microcoded version of the new Chrysalis process scheduler was tested during late February and early March using the BSAT message generator, local delivery and message sink processes. The Top-Level BSAT process was augmented with additional commands to collect and display BSAT performance statistics. These tests are described in greater detail in later sections.

During March, work began on the BSAT synchronous I/O device drivers. These routines will be used for the host interfaces and the interface to the ESI. Work also began during March on the internal loopback function which simulates some of the BSAT-ESI interactions.

During April, the host interface synchronous I/O code was completed and work began on the higher level Host Access Protocol (HAP) software. Most of the Host_In, Host_Out and HAP_Control processes were written. Work continued on the CPM-Xmit process which contained the synchronous I/O code for communicating with the ESI and transmitting bursts out over the satellite channel. As a result of the work on the host and channel synchronous I/O routines, performance problems were uncovered with the handling of the priority message queues. The mechanism by which these message queues are handled was redesigned to be much faster.

3.6 Performance Measurements of the BSAT and the Chrysalis Scheduler

The BSAT is an application that uses the Butterfly hardware and the Chrysalis operating system, and is composed of many processes on many processors. Because it is the most complex application program running on the Butterfly to date, we wanted to examine the operation of the BSAT to ensure it worked in practice as we had predicted it would in theory. We also wanted to gain a sufficient understanding of the system we were developing to predict what levels of performance are attainable and how they may be attained.

Any performance limitations of the hardware or operating system would impact directly the performance we could expect from the BSAT. A central facet of the Butterfly system is the Chrysalis process scheduler. If the scheduler were very slow, it would mean that the BSAT would have to be designed to minimize process switching and allow greater latencies in order to get the best overall performance. If scheduling did not tend to run the "appropriate" process at the "right" time, it would increase the complexity of the software as we tried to compensate for the scheduling deficiency.

As a result of our testing, several important modifications were made to the scheduling algorithm. After each modification, the tests were rerun. The end result is a scheduler that is fast, predictable, and understandable. We believe the schedulers will not be a factor limiting BSAT performance. The final form of the scheduler is given below, along with some background and definitions.

3.6.1 Scheduler Background

There is a separate instance of the process scheduler on every Processor Node. Each instance of the scheduler manages the processes that are local to that node.

Under Chrysalis, every process runs at one of four priority levels, with zero being the lowest. Process priority is initially established when the process is created. In addition to these four 'normal' priority queues, there is another set of four 'low' priority queues. If a process is using more than its fair share of time, the scheduler may temporarily move it down exactly four notches to the corresponding low-priority queue.

Process scheduling is based on the concept of an 'event.' The event mechanism is used when requesting a service, and, when the requested operation has completed, the server 'posts' the event to signal completion to the requesting process. In the meantime, the requesting process may continue to run, or it may choose to wait for the event by declaring itself non-runnable and asking the scheduler to wake it up when the event is posted. If the process continues to run, it may establish other events and test outstanding events for completion. If it decides to wait, it may wait for any outstanding event or for only one of a specific set of events.

In real-time systems it may be vital that a process get a certain minimum amount of CPU time every little while. The technique of epoch scheduling attacks this problem by defining a scheduling period called an epoch, which is currently 100 milliseconds. At the end of each epoch, a system process called

the Epoch Scheduler restores the scheduling state of all processes, moving each process to its normal-priority queue. Within the epoch, CPU time is allocated to processes based in part on a declared need. For example, if a need for 20 milliseconds out of each epoch is declared, then if the process is ready to run at the beginning of the epoch and remains ready throughout the epoch, the scheduler guarantees that the process will be allowed to run for at least 20 milliseconds.

Associated with each process is a minimum amount of time, called its "need", which is the required time that it must be allowed to run in each epoch. The sum of the declared needs of all processes on a given Processor Node is called committed time. Obviously, the scheduler cannot satisfy the needs of every process unless the amount of committed time is less than the total time available in an epoch. Time which is not promised is called uncommitted time. Committed time which is not used is called windfall. The Chrysalis scheduler is designed so that any uncommitted or windfall time is available to any process which needs it on a first-come-first-served basis (with preference given to higher priority processes). No process goes to low-queue until all uncommitted time has either elapsed or been allocated to specific processes. When a process on low queue is allowed to run, it is making use of windfall time.

In real-time systems, most processes normally run in relatively brief bursts, then wait for some event to trigger further activity. An epoch scheduler is well suited to this sort of behavior. Occasionally, some processes will want to run for extended periods, either due to some unusual occurrence or due to a greater-than-normal influx of data or service requests. In other applications, some processes will want to run continuously for relatively long periods.

To help with these cases, the scheduler provides each process with a time-slicing parameter. This parameter limits the time that a process can run without giving up control to other processes of the same priority. When the time slice of a running process ends, another process of equal priority will be scheduled on a round-robin basis. If this parameter is set too large, other processes may have to wait a long time for service and uncommitted and/or windfall time may not be fairly shared; if it is set too small, excessive scheduling overhead may be incurred.

The Butterfly processor maintains an interval timer that is used to support the time-slicing functions of the scheduler. When an interval timer interrupt occurs because the currently running process has reached the end of its time slice, the scheduler is run. This interrupt routine takes about 100 microseconds. It is expected that properly tuned processes will

normally dismiss voluntarily before their time slices end, so the impact of this extra overhead should be small in most cases.

Saving and restoring the process state is the most expensive part of the scheduler. The scheduler is organized to save the state of the currently running process only when it switches processes or goes into the idle state (when no process is runnable). The state of a process is restored only when a process switch occurs.

3.6.1.1 The Scheduler Algorithm

In addition to switching between runnable processes at appropriate times, it is up to the scheduler to ensure that each process gets its share of runtime, as dictated by its need and time-slice parameters.

If the machine is idle, posting any event will trigger the scheduler. Otherwise, posting an event owned by a process of higher priority than the currently running process will trigger the scheduler and preempt the current process. If the current time slice is exceeded, the timer interrupt routine will trigger the scheduler. Finally, if the process calls the Chrysalis functions Wait or MWait (directly or indirectly) and an appropriate event is not yet available, the process will be

marked not-runnable and the scheduler will be triggered.

The scheduler accounts for the time just used by the process and computes the next time slice the process should have. If the process is still runnable, it will be placed on the appropriate normal or low priority scheduling queue. Next, the priority queues are scanned to select the highest priority runnable process. Since the queues contain only processes which are runnable, the first process on the first non-empty queue is selected. If there are no runnable processes, the processor goes idle.

3.6.2 Scheduler Performance Measurement

Given the central role of the scheduler, it is important that its operation be both efficient and understandable. To achieve maximum performance, much of the scheduler and many related functions have been implemented in microcode. The Processor Node Controller (PNC) makes all of the necessary scheduling decisions and presents one of six interrupt vectors to the 68000, depending on what kind of context switch is required.

3.6.2.1 Measurement Tools

In order to observe the behavior of the scheduler and the application code, we installed a certain amount of instrumentation: the Process Control Block (PCB) was extended to make room for a set of per-process counters; the supervisor data segment (Segment F8) was changed to make room for a set of per-processor-node counters; and a small amount of code was added to the scheduler to maintain these counts. Since the impact of this instrumentation on performance appears to be minimal, we have left it in place to support future experiments.

The per-node counters include one counter for each of the six scheduler interrupt paths so that we can determine how often each path is taken. In the special path where the scheduler switches from one running process to another, we installed an additional counter that records how often the switch is caused by a high queue process interrupting a low queue process. All of these counts are cumulative from the time Chrysalis first begins running.

In the Process Control Block definition, there was already a record of the total run time of each process being maintained by the Epoch Scheduler. We added to this a count of the number of times the process has been scheduled to run. Both of these counts are cumulative from the time that the process was created.

A collection of software procedures was added to the BSAT to sample all of the counters in the system and present a readable summary of their contents. The numbers are scaled to convenient units, such as milliseconds of run time per second of real time, or events per second, before being displayed. The procedures also allow the sample interval to be varied easily so that the experimenter can see both short term and longer term behavior of the system.

3.6.2.2 Method of Measurement

The BSAT processes used in the testing consisted of a Message Generator, two types of Message Sink, and a Delivery (routing) process. The operation of these processes was well understood and could be easily modelled. These processes can be easily controlled and moved between processor nodes. The BSAT monitoring and display process was run in a separate processor from the processes being measured so that it would not interfere with the measurements.

The principal method of analysis was one of making incremental changes to the system under test and observing the resulting changes in its behavior. The changes consisted of varying the frequency and type of scheduling performed and the amount of uncommitted time in the epoch. The behavior consisted

of the number of packets sent through the system (a measure of useful work done), the percentage of the CPU used by each process, and the number and type of schedulings that occurred.

A process was modelled as having two components: time spent executing the application task, and time spent doing system overhead functions. Timer interrupts, for example, are charged to the then-running process as far as scheduling is concerned, though the interrupted process gets no useful work done during the interrupt. Similarly, time spent saving and restoring the process state is charged to a process, even though the application process is not progressing during that time. Observing changes in the number of packets transmitted was used as a measure of time spent executing the application task. Changes in the charged run time of a process, coupled with knowledge of time spent executing the application work, may be used to measure time spent in a system overhead operation.

The foregoing may be summed up in the following equation:

$$t = M * B + n * s \qquad (1)$$

where:

t = total run time needed (msec. CPU time/1 sec. clock time)

M = messages/second flowing through the system (messages/sec)

B = time needed to process one message (msec./message)

n = number of times the process is scheduled (1/second)

s = overhead time per scheduling (msec.).

This equation may be solved for "s" to yield:

$$s = (t - M * B) / n \qquad (2)$$

This says that the average time spent per scheduling is the total run time of the process minus the time spent doing the application's work, spread over the total number of times that the process was scheduled.

From the experimental data we can get values for t, M and n directly. This leaves the problem of determining B and s. This problem is solved by considering the equation as having the unknowns B and s, and declaring that these values must be constant from one set of experimental data to the next. Using this assumption, we can take two sets of measurements and use the following formula (derived from equation 1) to estimate B:

$$B = (n2 * t1 - n1 * t2) / (n2 * M1 - n1 * M2) \qquad (3)$$

where t1, M1, and n1 are from one experiment and t2, M2, and n2 are from a second measurement. The value of B is milliseconds of run time per message processed.

It turns out that calculation of "s" using the formula above is very sensitive to small changes in B. We were thus able to

determine the value of B with good accuracy.

It should be noted that the time per scheduling(s) that we compute from these measurements is NOT the same as the microcode scheduling time, it is the total of anything that occurs when the process is scheduled and may include timer interrupts, Chrysalis system code invoked only when a process stops or starts, or user process code invoked only when the process awakens or goes quiescent. In interpreting the measurements, it was necessary to identify all the various causes of once-per-scheduling time spent before we could believe that we understood the behavior of the scheduler.

3.6.2.3 Experimental Results

In the experiments that we conducted, we quickly determined that the Chrysalis system processes behaved in a constant and experimentally uninteresting fashion. In brief, the Epoch Scheduler caused 10 preemptive process swaps per second, and another 10 process swaps as it returned to running the lower priority process it had preempted. The Remote Demon, responsible for garbage collecting memory, also caused 20 process swaps per second. Together, they took a relatively constant 3.1% of CPU time. Most of this is taken by the Remote Demon, whose run time is a parameter that we can change.

Because these two processes are so constant, the experiments below are described as if they were not present. However, each processor always had an Epoch Scheduler and a Remote Demon running on it.

The other near-constant we found is that 0.8% of the CPU time is not charged to any process. We believe that most of it is lost between the time one process ceases to be charged and the next process starts to be charged. Some of it may also be lost because of the manner in which the Epoch Scheduler attempts to update its own run time while it is running.

In the first experiment we determined the time needed to reschedule a process. To do this, the Message Generator was started with its parameters set so that it would try to send to an illegal destination. In this mode it will run constantly. The Message Generator was the only process trying to run on its processor.

The process requested 100% of the CPU time, so the processor was overcommitted. By doing this, the Message Generator was guaranteed never to be placed on low-queue. The experiment consisted of varying the scheduling slice parameter from 96 milliseconds per slice (one scheduling per epoch) to 1 millisecond (approximately 96 schedulings per epoch).

The time to re-schedule a process whose time slice has run out was measured at 129 microseconds. This includes the time spent in the timer interrupt code, which also implements a wakeup service that is not directly related to the scheduler. In later experiments we found that the timer interrupt code seems to take about 100 microseconds to run. This means that the microcode time for stopping and restarting a process is approximately 29 microseconds.

In the next experiment the Message Generator was set up to send messages via a Dual Queue to a Message Sink process in the same processor. Both processes always ran on high-queue. The Message Generator ran until its time slice had run out, while the Message Sink voluntarily dismissed when it found it had no more messages to discard. Both processes ran at the same priority and so were round-robin scheduled by the Butterfly scheduler. Again the scheduling slices were varied.

We found that the time to switch from the Message Sink to the Message Generator was 119 microseconds. The time to switch from the Message Generator to the Message Sink was 219 microseconds. The difference between these two types of context switch is the 100 microsecond runtime of the timer interrupt routine.

In another experiment we varied the amount of committed time requested by the Message Generator as well. This meant that during part of each epoch the Message Generator ran on low-queue. Since the Message Sink always ran on high-queue, and since Enq_DualQ to a higher priority process results in an immediate preemption, we were able to cause the scheduler to be invoked for every message and to measure the time spent during preemptive scheduling.

We found that the total time for a preemptive process swap was 135 microseconds. However, this measurement includes the time that it takes to post the Event that triggers the preemption. Thus, we can separate this result into 119 microseconds for the swap and 16 microseconds for the Posting of the event. Our experiments are not sufficiently complete to be sure that this division of the 135 microseconds is the correct one, but it corresponds closely to what we expected.

The results of these experiments are summarized in Table 1. These numbers appear to be consistent with what we know about the operation of the Butterfly hardware. For example, the difference in time between switching processes and restarting a process is mostly the time to save and restore the registers when switching. The difference of 119 - 29 = 90 microseconds is slightly (and within experimental error) larger than the time a 68000 takes to

SCHEDULER TIMINGS

OPERATION	TIME	COMPONENTS
RESTART PROCESS	129 μsec	29 μusec MICROCODE 100 μsec TIMER INTERRUPT
ROUND-ROBIN SWAP AT END OF SLICE	219 μsec	119 μsec REGISTER SWAP & MICROCODE 100 μsec TIMER INTERRUPT
ROUND-ROBIN SWAP ON VOLUNTARY DISMISS	119 μsec	REGISTER SWAP & MICROCODE
PREEMPTIVE SWAP	135 μsec	16 μsec EVENT POSTING 119 μsec REGISTER SWAP & MICROCODE

Scheduler Performance Measurement Results
Table 1

perform a register save and restore.

In addition to these experiments, we were able to make some preliminary measurements on the BSAT processes. Four types of process were used. There was a Message Generator Process which composed messages and put them onto a Dual Queue. There were two kinds of Message Sink process. The "simple" Message Sink received message IDs on a Dual Queue and discarded them. When a message ID appeared, the Simple Message Sink removed it from the Queue, mapped in the message, incremented a counter, and freed the message buffers. The "complex" Message Sink simulated part of the activity of a BSAT host I/O process, and its processing rate was much lower (in fact, the results of these measurements prompted a rewrite of part of the corresponding BSAT process). We also used the BSAT Local Delivery process. The function of this process in the BSAT is to route messages that are destined for locally connected hosts. When a message ID arrives, the Local Delivery Process removes the ID from a Dual Queue, maps in the message, uses the header to make a routing decision, enqueues the message ID to the appropriate host output queue, and posts an event.

Some preliminary throughput measurements for these processes are given in Table 3. The first part of the table shows the processing rate of the Message Generator, the Simple Message

Processing Rate

Process	Messages/second	msec/message
Message Generator		
queue on same processor	1811	0.5522
queue on remote processor	1633	0.612
Simple Message Sink	6580	0.1520
Local Delivery	2095	0.478

Combined Throughput - Single Processor Node

Processes	Messages/second	msec/message
Message Generator/ Simple Message Sink	1345	0.743
Message Generator/ Local Delivery/ Complex Message Sink	688	1.453

Table 2. Preliminary BSAT Throughput Measurements

Sink, and the Local Delivery process, running on separate Processor Nodes. For the Message Generator, results are given for two cases. In one case, the Message Generator output queue was local to the process. In the other case, the queue was on a remote node and the enqueue operation typically involved Posting an event to a process waiting on the queue. This accounts for the difference in throughput. The second part of Table 3 gives

the measured throughput for the Message Generator, both types of Message Sink, and the Local Delivery Process running together on the same Processor Node. As expected, these numbers indicate lower throughput.

Since these measurements were made, some of the BSAT code has been rewritten to be faster. Measurements of Chrysalis routines have changed our understanding of which operations are "expensive" and which are "cheap," and will be a basis for tuning Chrysalis, the Voice Funnel and the BSAT for better performance.

Another aspect of the scheduler that was tested but which is difficult to describe easily is the handling of committed, uncommitted, and windfall time. We performed tests that determined that the scheduler does indeed run processes on high-queue during uncommitted time, moves them to low-queue at the proper point after committed time has begun, and will properly preempt lower priority processes when a higher priority process becomes runnable. These changes were seen as increases both in the number of times selected processes were scheduled, and in the throughput numbers. By modelling the processes' behavior during each phase of the epoch (uncommitted, committed, windfall), we were able to compare our composite numbers with the observed values. There was excellent agreement.

3.6.3 Conclusions

We have developed a set of process measurement tools that will be very valuable in determining and tuning system performance. This monitoring and tuning includes the problem of load balancing among the individual processors. More importantly, we can measure the performance of individual processes in the BSAT to determine bottlenecks in processing and to determine the process's inherent throughput.

Our measurements so far have encouraged us to believe that message processing rates in excess of 1000 messages per second are attainable, and that 1500 messages per second may be possible in a basic, but well tuned, system. With the notable exception of the Channel Scheduler process, most of the BSAT processes could be duplicated and run in parallel on additional processor boards to multiply throughput if desired.

4 INTERNET OPERATIONS AND MAINTENANCE

This QTR is the final Internet report written for the current contract. Internet work is being continued on contract MDA 903-83-C-0131.

The major activity during the past quarter was the continued deployment and maintenance of the Macro-11 gateway. New gateways are now running at ISI (Arpanet - Ethernet) and Rockwell (Arpanet - PRnet). The current list of operational gateways is shown in the following table.

Gateway	Adjoining Networks
BBN	ARPANET - BBN-NET
BBN-PR	BBN-NET - BBN-PR
BRAGG	ARPANET - BRAGG-PR
BRIMF	ARPANET - DEMO-PR
CSS	ARPANET - SATNET
CRONUS	ARPANET - DOS-ETHERNET - FIBERNET
DCEC	ARPANET - SATNET - EDN
ISI	ARPANET - ISI-ETHERNET
NTARE	SATNET - NTARE-TIU - NTARE-RING
PURDUE	ARPANET - PURDUE-NET
ROCKWELL	ARPANET - ROCKWELL-PR
SRI-C3P0	ARPANET - SF-PR-2
SRI-C3PR	ARPANET - SF-PR-3
SRI-R2D2	ARPANET - SF-PR-1
UCL	SATNET - UCLNET - RSRE/NULL - UCL-TACNET
WISC	ARPANET - WISC-NET

A new gateway software release (1004.) is now running in the majority of gateway sites (binaries have been delivered to the remainder). The release was delayed somewhat because we discovered a bug in the routing code (GGP) dealing with sequence

numbers wrapping around. The bug, which was also in the previous releases, is now fixed. The new release contains the following additions and improvements:

- o Arpanet "Uncontrolled" messages supported.
- o ICMP Source Quence Sent.
- o Class-C Arpanet Addressing supported.
- o New Gateway Buffer Allocation.
- o Interface and Neighbor probes sent with Higher priority.
- o Satnet Support Improved.
- o CMCC Removed.
- o Interface Up/Down Procedures (K out of N) Improved.
- o Ethernet supported (fixed table addess lookup).
- o BBN Fibernet supported.
- o HMP Status Reports show Memory Useage.
- o Bug fixes and improved Traps.

The MOS driver for X.25 Interface for the VAN gateway is now complete. We expect to begin testing with the Telenet line in the next month.

5 MOBILE ACCESS TERMINAL NETWORK

As part of our participation in the development of the Mobile Access Terminal (MAT) and the MAT Satellite Network (MATNET), we field tested Satellite IMP version 6.2:4, which incorporated a change in the satellite channel scheduling so as to factor in the message attribute PRIORITY only; accordingly, channel allocation is more consistent with the preferences of the Navy. We designed, assembled, tested, and released Satellite IMP version 6.2:5, which incorporated major changes in the packet header formats to eliminate the satellite receive channel blockage problem observed during January's contention tests. We provided support for the system integration and the ongoing large scale system testing, which included participation by E-Systems, ECI Division, in St. Petersburg, Florida; Tracor, Inc., in San Diego, California; and the Advanced Command and Control Architectural Testbed (ACCAT) at the Naval Ocean Systems Center (NOSC) in San Diego, California. These major tasks are detailed in the following sections, while some of our other activities are described immediately below. In the MATNET project, the Terminal Input Unit (TIU) hardware, the COMSEC equipment, the Black processors, and the radio equipment are ECI's responsibility, while the C/30 Satellite IMPs, the gateway, and the TIU software are BBN's responsibility.

Upgrading of the MATNET testbed C/30 Satellite IMP hardware was completed, where upgrading included conversion from a 3-slot to a 7-slot chassis, installation of a larger power supply, replacement of the power connectors with connectors of the locking type, replacement of the power cables, and replacement of the 16 MHz master crystal to one with more accuracy. Because the hardware was outmoded, the upgrading process was accompanied by considerable aggravation. Afterwards, we implemented all the modifications necessary to enhance survivability in a sea-going environment (see BBN Combined Quarterly Technical Report No. 27). This unit, which was renamed Satellite IMP #5 consistent with its inclusion in the SHIP5 site, was tested at BBN and subsequently shipped by air freight to ECI for integration with associated Black equipment and TIU.

We developed Satellite IMP macrocode and TIU software patches which allow two abutting MATs to communicate with each other via their 1822 Host-to-IMP interfaces, even though the MATs are configured on two separate independent satellite networks. In order to provide the correct packet routing, one of the two Satellite IMPs incorporates a software module performing the function of a rudimentary gateway. The patches were tested at ECI with Satellite IMPs #4 and #5 terrestrially linked, using a special cable we procured for interchanging the transmit and receive data and control lines. At this time, user TCP

connections between TIUs on different satellite networks were successfully demonstrated.

Among the status display lights on the C/30 front panel is an indicator designated to represent the looped status of the satellite channel. In operation, the Satellite IMP determines the round-trip time of a packet sent over the satellite channel and declares the channel looped if the round-trip time is below a threshold value. Before release of Satellite IMP version 6.2:5, the satellite channel looped status display indicated only when a Satellite IMP was in internal crosspatch mode or was looped through the digital satellite simulator but not when the Black processor was set to a looped state. In Satellite IMP version 6.2:5, we changed the macrocode to have the satellite channel looped status display indicate when any of these situations occurs. Because Black processors are remotely located from their associated Satellite IMPs, operational procedures should benefit from this modification.

We assembled, tested, and released Satellite IMP version 6.2:6, which incorporated several new traps specifically to aid testing and which fixed several bugs occurring in version 6.2:5. User selected optional traps were added, so that console terminal printout could be obtained during each monitoring interval (slightly less than three minutes) for the following: the total

number of Channel Allocation Request packets transmitted by this site; the total number of Channel Allocation Request packets received from this site; the total number of Channel Allocation Request packets received from all other sites; and the total number of Hello packets received from all sites. When contention packet testing is undertaken, the above information indicates how well packets are being received. By collecting this information on each site's console terminal, we avoid having to depend on the remote sites achieving reservation synchronization in order to transmit monitoring reports to the Network Operations Center via the SHORE1 site at NOSC.

In Satellite IMP version 6.2:6, we fixed a bug that sometimes prevented an internally crosspatched Satellite IMP from achieving frame synchronization, because the initialization procedure for the USART in the satellite I/O interface generated satellite channel round-trip times less than zero. Now negative round-trip times, although identified by the generation of a specific trap, are ignored in the scheduling algorithm; necessarily, macrocode modules to acquire and to maintain frame synchronization were modified. Also fixed is the configuration table used by the internal gateway code; incorrect table entries prevented us from connecting the SHIP2 site directly to the PLI at NOSC last January. Lastly, we fixed a message generator code bug which prevented the generation of uniformly distributed

message lengths; previously only packets having the maximum length specified were generated.

Conversion of MATNET from a Class A net to a Class B net has been initiated by the Internet Configuration Control Board. Whereas MATNET was originally assigned the 8-bit Class A net number 34, the new assignment is the 16-bit Class B net number represented by the octet pair 128.013. In preparation for running two independent MAT satellite networks on two different FLTSATCOM satellites, we procured a second Class B net number represented by the octet pair 128.017. To implement the net number change, we have to alter the network address switch positions on the TIU robustness card as well as modifying the Satellite IMP macrocode, the gateway software, and the TIU software. Because SRI International has not released their new TIU software written in C language and accommodating Class B net numbers, we have delayed implementation of the new MATNET Class B net numbers. Only when MATNET interacts with the rest of the unclassified internet system will this be a problem.

In one of last January's experiments, we observed that the Black processor data indicated that the SHIP2 site received 7% more unique words than were transmitted (see BBN Combined Quarterly Technical Report No. 28). Since February's experiments were unable to duplicate this discrepancy, further examination of

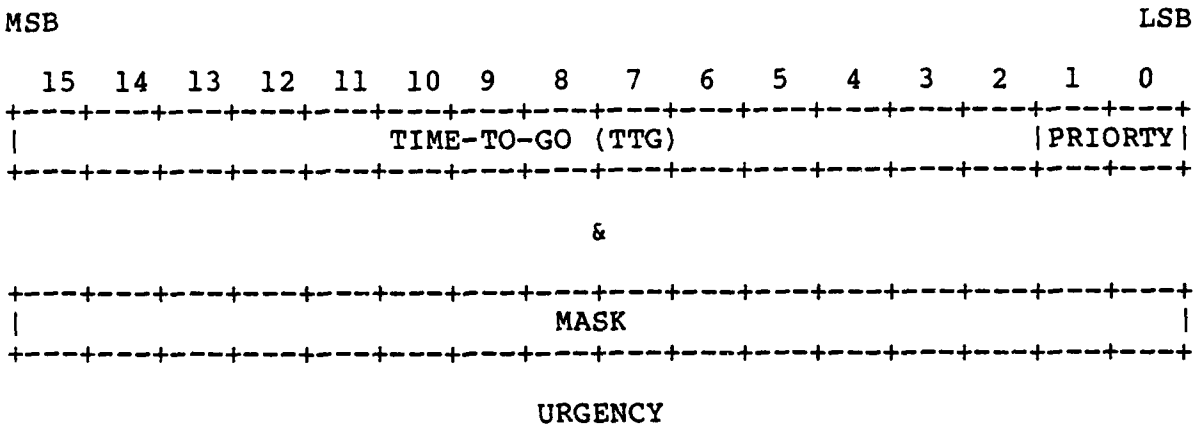
the equipment leads us to believe that the discrepancy is due to faulty seating of cards in the Black processor and not due to a fundamental system design deficiency. The SHIP2 Black processor has been returned to ECI for refurbishment.

5.1 Satellite Channel Scheduling

In SATNET and subsequently in MATNET, a message has two independently specified attributes which are taken into account for the scheduling of its transmission over the satellite channel: namely, DELAY-CLASS and PRIORITY. The former, which is a measure of the timeliness of a message, corresponds to the acceptable time span for a message to remain within the system. (Although it is undesirable for a message to remain undelivered to the destination host when the time span indicated by the DELAY-CLASS value has expired, a third message attribute HOLDING-TIME determines the actual time when the message is declared undeliverable and is expunged from the system.) To represent when the message time span expires, the message Time-to-Go (TTG) is defined as the time of the message delivery from the source host to the source Satellite IMP plus the message's DELAY-CLASS value in units of 40.96 milliseconds. DELAY-CLASS has the values {1, 2, 5, 20} seconds, where user interactive traffic is expected to assume the smaller values, and bulk data

traffic is expected to assume the larger values. The second message attribute PRIORITY is expressed as a 2-bit quantity to specify four levels {0-3}, where the value 0 signifies the highest priority.

TTG and PRIORITY are catenated into one 16-bit word and are logically ANDed with a 16-bit quantity designated as MASK to form the URGENCY of a message as shown below.



MASK is a weighting mask for adjusting the relative importance of TTG and PRIORITY in determining the message URGENCY. In SATNET and formerly in MATNET, MASK has all bits on (MASK value equals -1), so that URGENCY contains all the information in TTG and PRIORITY. Channel scheduling is done by serving the most urgent messages (lowest value of URGENCY) first; thus, as messages remain longer within the system, they get preferential treatment. Because PRIORITY is significant only for messages having equal TTGs (after being masked by MASK), PRIORITY has the lesser

significance of the two attributes. Consequently, user interactive traffic of lower PRIORITY can have preferential treatment over bulk data traffic of higher PRIORITY. For messages of equal URGENCY, channel scheduling incorporates a round robin service among all sites with a FIFO (First-In, First-Out) service to any individual site to promote fairness.

It was originally thought that the bits to be masked out, if any, would probably be the low order bits of the TTG field; then more TTG ties would occur and PRIORITY would have greater importance. In MATNET Satellite IMP version 6.2:4, however, we carried this concept to its extreme by changing MASK to mask out all bits of the TTG field, leaving only the PRIORITY bits in the calculation of URGENCY (MASK value equals 3). Channel scheduling is therefore by PRIORITY only with DELAY-CLASS having no bearing at all. Within each PRIORITY level, the round robin service among all sites with FIFO service to any individual site enforces fairness. Satellite IMP version 6.2:4 by virtue of its changed channel scheduling is incompatible with all previous versions.

Results of satellite tests of Satellite IMP version 6.2:4 software using message generators simulating dense traffic in all sites simultaneously demonstrated that fairness is superior in MATNET relative to SATNET; in MATNET no one site can dominate the channel, whereas in SATNET the channel can be dominated (although

not captured) by heavy users. The latter effect occurs because sites with considerable traffic have more opportunities to make reservations than sites with little traffic due to piggybacking of reservations for channel time on previous messages already having entries in the scheduling queue. Channel scheduling with TTG factored in favors the existing heavy user at the expense of a light user trying to increase his share of the channel bandwidth.

5.2 Packet Header Format Changes

The major change in Satellite IMP version 6.2:5 is that we revamped the packet header formats in order to leave space for extra error protection bits on the data traversing the satellite channel. The goal was to eliminate the satellite receive channel blockage problem (see next section) which occurs whenever the satellite channel is extremely noisy or packet contentions occur. We also restructured the common header part of all MATNET satellite channel packets to increase the SOURCE-SIMP address field from 3 bits to 5 bits; hence, the limitation in the address field to future expansions of MATNET is a maximum of 31 sites instead of 7 sites. Satellite IMP version 6.2:5 by virtue of these packet header changes is incompatible with all previous Satellite IMP versions as well as with all previous Black

processor firmware releases. Since these changes affect the format and timing of data transfers between the Satellite IMP and the Black processor, modifications to the Satellite IMP in both the satellite I/O microcode and macrocode were necessary.

Increased error protection on the packet size parameter and the compressed message indicator in the Black processor are essential to eliminating the satellite receive channel blockage problem. Unfortunately, the original channel allocation request packet format was unable to accommodate the extra bits necessary for the added error protection without exceeding the capacity of a single interleaver block. Not wanting to increase the size of this packet and thereby increase the size of the entire control subframe by 100%, we chose to remove the space for one of the two message reservations in this packet header, automatically freeing 32 bits for reallocation in the system. Consequently, a site has fewer opportunities to procure channel bandwidth for accommodating its traffic. To offset the impact of this restriction, we left the data packet headers with space for two message reservations. Thus, when a site piggybacks reservations for channel time on previous messages already having entries in the scheduling queue, that site can increase its share of the satellite channel bandwidth geometrically. Although satellite channel data packets have increased in size by 32 bits due to the extra bits added for error protection, the maximum length data

packet, containing 128 host user data words, continues to fit within a 10 interleaver block packet.

Space totaling 32 bits was allocated for error protection in the system as follows: 8 bits to increase protection for the Black processor packet size parameter; 16 bits to generate protection for the Black processor compressed message indicator, which never had protection before; and 8 bits to increase the length of the Satellite IMP packet start framing sequence. For the latter, an ASCII STX character was added to complement the ASCII SYN character already being used. In the Satellite IMP, the satellite I/O microcode was modified to affix the STX character after the SYN character when transferring packets to the Black processor and to check for a following STX character after each detected SYN character when receiving packets from the Black processor. If the SYN-STX combination is found, the microcode checks the validity of the following packet length field (see BBN Combined Quarterly Technical Report No. 27). Only when a proper SYN-STX combination is found and the packet length passes the parity check is a valid packet start framing sequence declared. If either check fails, the microcode increments the FALSE-SYN counter and resets the USART in the satellite modem interface into SYN-hunting mode without simulating a DMA transfer to the macrocode. The STX character insertion/deletion has been implemented in the Satellite IMP microcode in such a fashion as

to be transparent to the Satellite IMP macrocode.

Due to the packet header changes, the number of bits transferred from the Satellite IMP to the Black processor (designated Transmit Data In or TDI bits) for each packet size (1-10 interleaver blocks) and for each coding state (rate 1/2 coding or no coding) was reduced from its previous value by 24 bits. All TDI counts now correspond to an odd number of bytes instead of an integral number of words. Shown in the table below are the TDI counts for packet sizes 1-10 interleaver blocks and rate 1/2 coding. Also included in the table are the number of host user data words and the total number of words processed by the Satellite IMP macrocode (including the 14-word MATNET data packet header) for traversing the satellite channel.

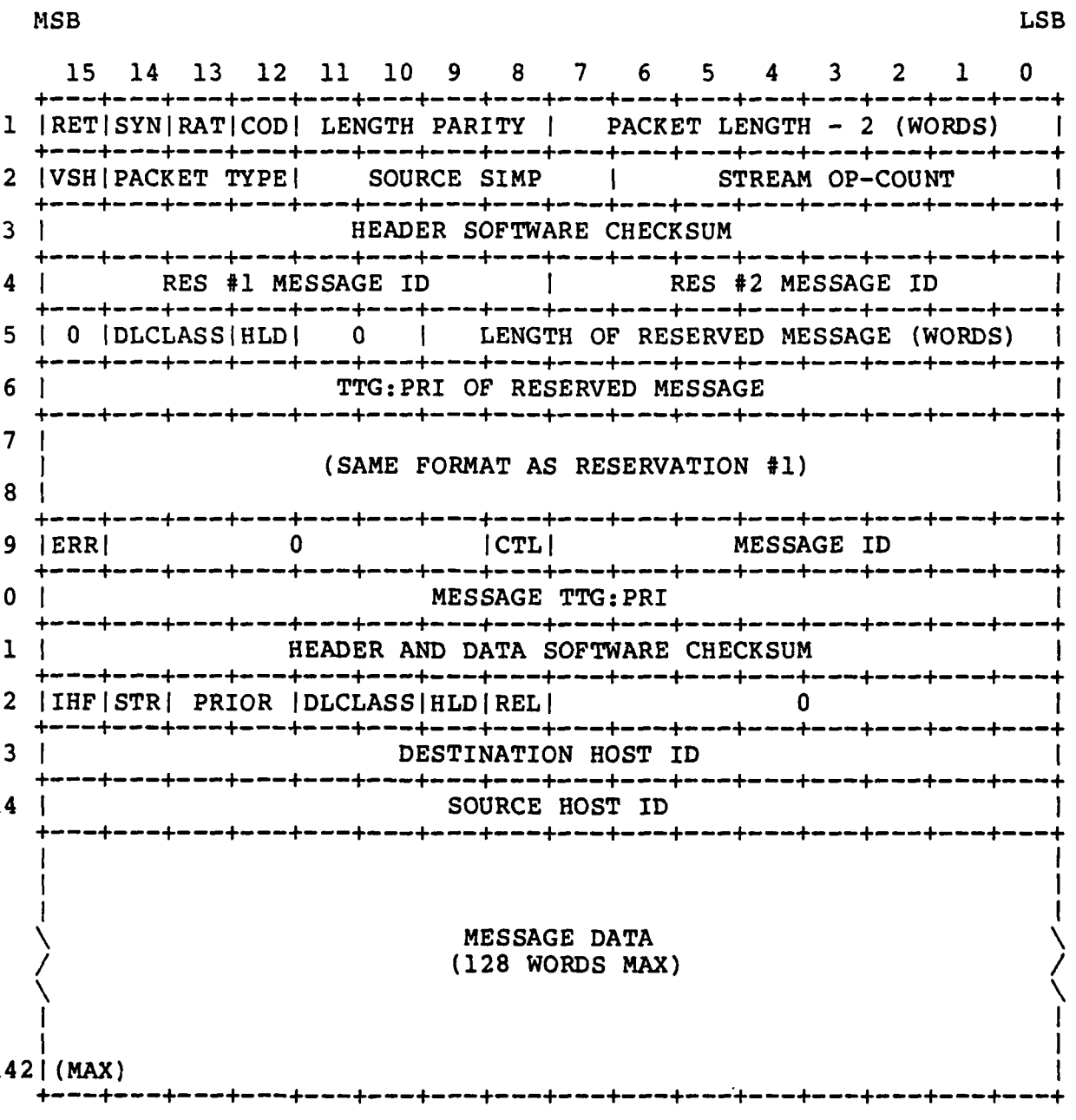
PACKET SIZE (blocks)	TDI (bytes)	USER DATA (words)	TOTAL (words)
1	125	0	0- 6
2	157	1- 8	7- 22
3	187	9- 23	23- 37
4	219	24- 39	38- 53
5	251	40- 55	54- 69
6	283	56- 71	70- 85
7	313	72- 86	86-100
8	345	87-102	101-116
9	377	103-118	117-132
10	409	119-128	133-142

Satellite IMP to Black Processor Transfers

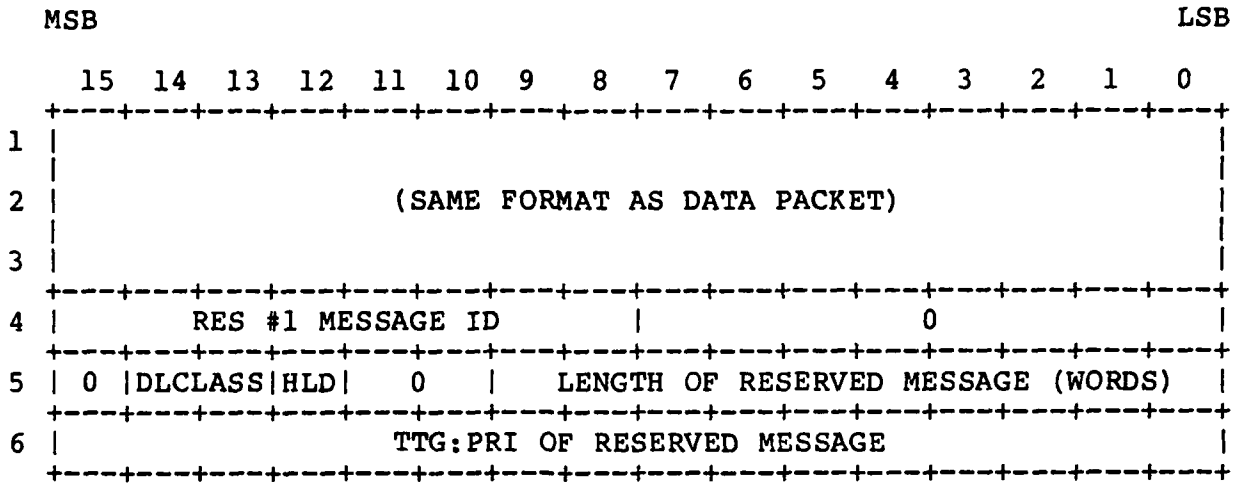
In order to implement the new TDI values, three tables in the satellite I/O module of the Satellite IMP macrocode were changed. The first, specifying the transmit DMA buffer size for each packet size and coding state, had each entry reduced by two words. The second, specifing the number of bits available to the macrocode in the first block of a packet for each coding state (used to calculate the number of interleaver blocks required for a packet of a given length), had each entry reduced by 32 bits. The third, specifying the length of the "DEBUG" (or "TPA") pulse used for enabling the Transmit Data clock when testing the system with the Digital Satellite Simulator, had each entry reduced by an interval corresponding to 24 bit-times at the TDI transfer rate of 51.2 Kbps.

Also appropriately redefined were the number of bits transferred from the Black processor to the Satellite IMP (designated Receive Data Out or RDO bits) for each packet size and for each coding state. Since the new packet formats caused some small changes in the receive-side timing across the Red/Black interface, parameter adjustments were required in the Satellite IMP macrocode defining the time window used for matching the SYN character receive time of a packet with the time of a Unique Word interrupt.

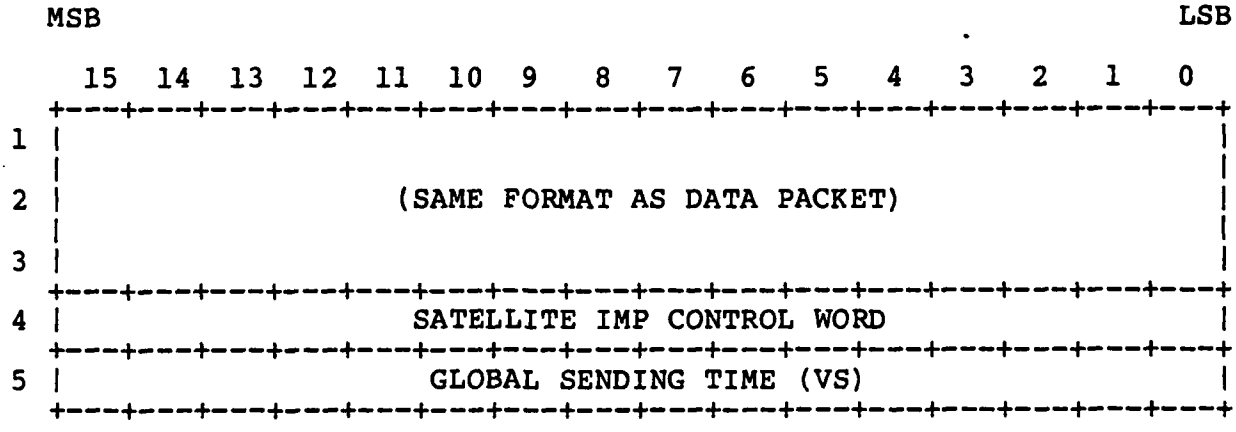
Shown below are the formats of the different types of satellite channel packets as generated and received by Satellite IMP version 6.2:5; depicted is the information between the STX character and the cyclic redundancy checksum (CRC) inserted by the microcode.



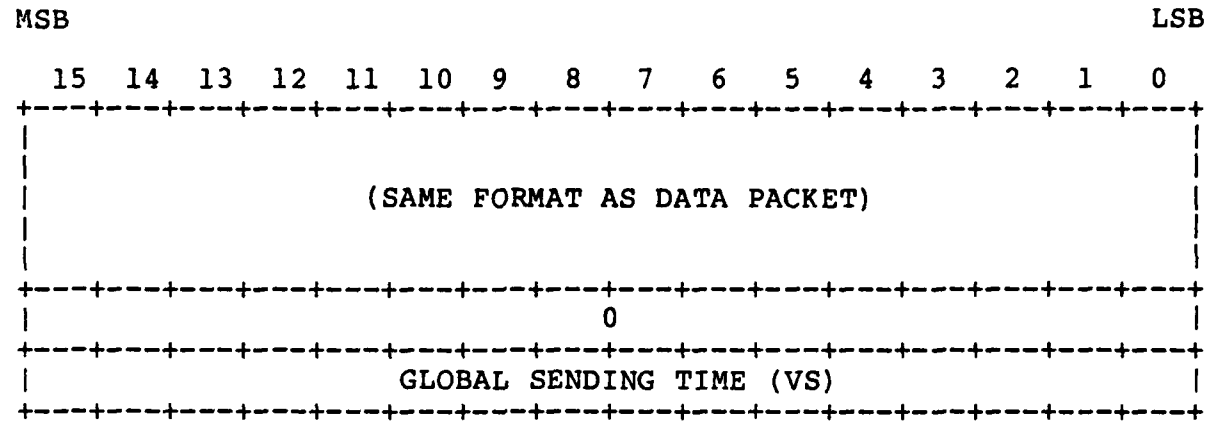
MATNET Data Packet



MATNET Channel Allocation Request Packet

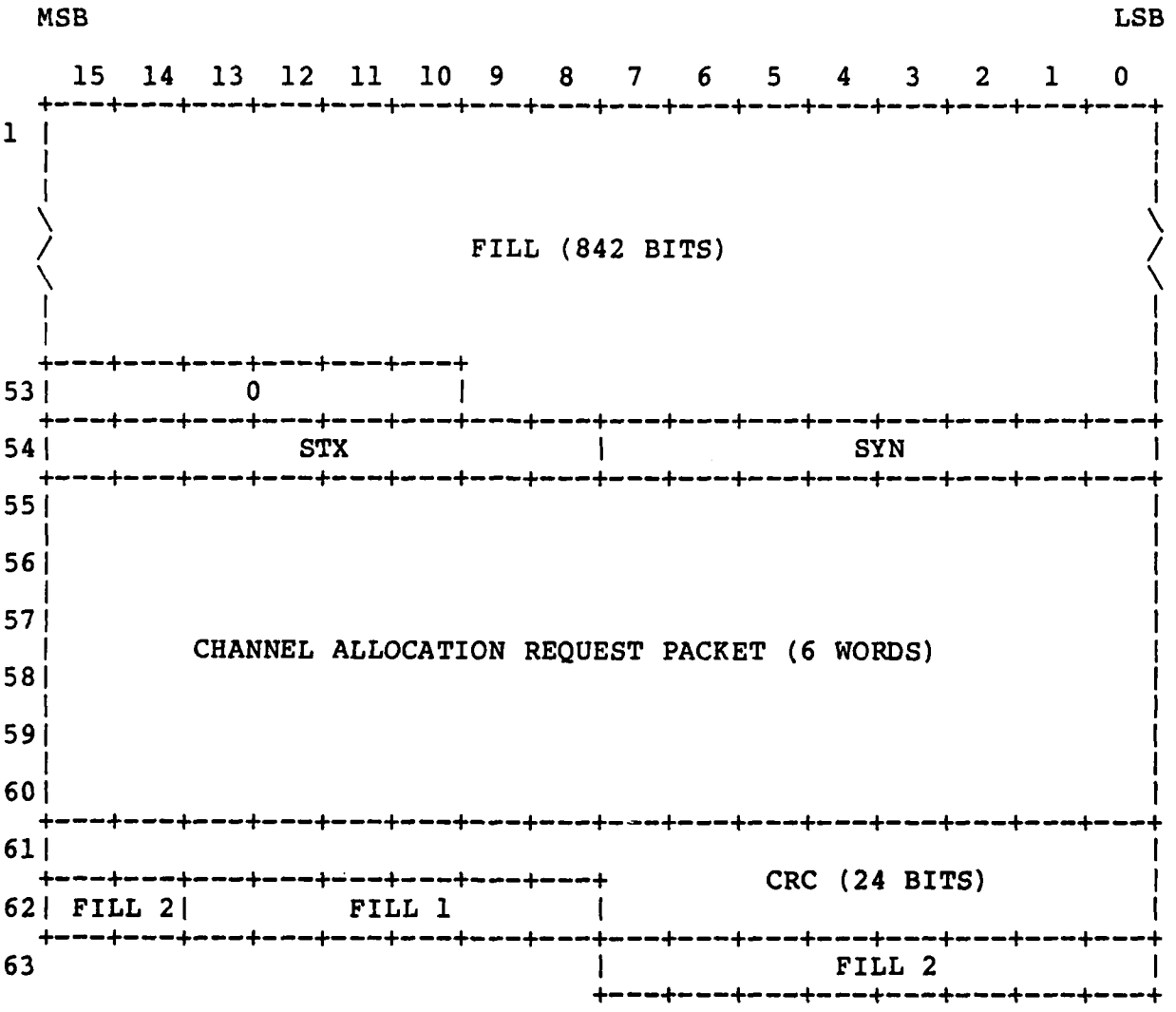


MATNET Hello Packet



MATNET Round-Trip Time Packet

Below is an example showing a channel allocation request packet bit transfer from the Satellite IMP to the Black processor (LSB first). Inserted into a single interleaver block are all of the bits from bit 10 of word 53 through bit 13 of word 62. The six zero bits transmitted before the SYN character allow the microcode to do bytewise processing of the macrocode DMA buffer; the bits labeled "FILL 1" are required to fill out the first interleaver block; the bits labeled "FILL 2" include flush bits and bits required to round the transfer up to an odd byte boundary.



Satellite IMP to Black Processor (TDI) Transfer Format
for a Channel Allocation Request Packet

5.3 Satellite Receive Channel Blockage

The contention packet testing undertaken in January (see BBN Combined Quarterly Technical Report No. 28) revealed that the implementations of the MATNET Red and Black subsystems were unable to provide reasonable system performance when contention packets occurred on the satellite channel. Network performance was so severely degraded, apparently due to the inability of the system to correctly determine packet boundaries of contention packets, that synchronization of all sites could not be maintained. Because of the far-reaching impact of this conclusion on the MATNET project, we repeated these experiments in February to confirm January's test results. Independent of whether common transmit and common receive antennas or separate transmit and common receive antennas were used by the SHORE1 and the SHIP2 sites at NOSC, the system, while stable when the sites transmitted Channel Allocation Request packets in different control slots, became unstable when the sites transmitted Channel Allocation Request packets in the same slot.

In the experimental setup for contention packet testing, the AN/WSC-3 transmitter for each site is first checked for nominal power and frequency settings, and each Black processor's quality monitoring threshold is set to its maximum value, effectively disabling the quality monitoring circuitry. After the usual

Black-subsystem-only satellite channel tests are performed at each site, MATNET is brought up running the FPODA satellite channel scheduling protocol. In order to generate recurring collision traffic, each Satellite IMP's channel protocol module is patched to transmit a Channel Allocation Request packet in the same control subframe slot in every PODA frame, independent of whether there are any reservations or acknowledgments to be sent. All normal traffic is removed from the channel by disabling the generation of MONITOR reports in both sites and by halting the MATNET gateway. Although the January and February tests with Satellite IMP version 6.2:4 showed frame synchronization was continually lost by both the SHORE1 and SHIP2 sites, several hours of tests with Satellite IMP version 6.2:5 and the new Black processor PROMs showed no loss of frame synchronization in either site. Therefore, we believe the increased error protection on the Black processor packet size parameter and the Black processor compressed message indicator, as well as the more stringent Satellite IMP packet start framing sequence, have fixed the satellite receive channel blockage problem.

Accompanying these tests, however, were some minor problems that consumed valuable satellite testing time. At NOSC, a crypto mode switch was set wrong and later a crypto loading operation failed. Switch positions on the DMA card in the SHIP2 Black processor were altered during shipment from ECI to NOSC. The

Black processor firmware had a bug, which caused a small percentage of 9 and 10 interleaver block packets to be dropped in the Red/Black interface. An ACCAT TOPS-20 crash destroyed our test records. Lastly, early in the tests, the ACCAT TOPS-20 would not respond to open connection negotiations from the TIU's TCP. We were uncertain at the time whether Satellite IMP version 6.2:5 had a bug causing the malfunction, so Satellite IMP version 6.2:4 sofware was loaded, but the symptoms were the same. Since the problem mysteriously cleared up the next day, we attribute it to a vagary in the TOPS-20.

DISTRIBUTION

ARPA
Director (3 copies)
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209
Attn: Program Manager
R. Kahn
R. Ohlander
D. Adams
B. Leiner

DEFENSE DOCUMENTATION CENTER (12 copies)
Cameron Station
Alexandria, VA 22314

DEFENSE COMMUNICATIONS ENGINEERING CENTER
1860 Wiehle Road
Reston, VA 22090
Attn: Lt. Col. F. Zimmerman

DEPARTMENT OF DEFENSE
9800 Savage Road
Ft. Meade, MD 20755
Attn: R. McFarland C132 (2 copies)

DEFENSE COMMUNICATIONS AGENCY
8th and South Courthouse Road
Arlington, VA 22204
Attn: Code B645
Glynn Parker, Code B626

NAVAL ELECTRONIC SYSTEMS COMMAND
Department of the Navy
Washington, DC 20360
Attn: B. Hughes, Code 6111
F. Deckelman, Code 6131

MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02138
Attn: D. Clark

MIT Lincoln Laboratory
244 Woods Street
Lexington, MA 02173
Attn: C. Weinstein

DISTRIBUTION cont'd

BOLT BERANEK AND NEWMAN INC.
1300 North 17th Street
Arlington, VA 22209
Attn: E. Wolf

BOLT BERANEK AND NEWMAN INC.
10 Moulton Street
Cambridge, MA 02238

S. Blumenthal
M. Brescia
R. Bressler
R. Brooks
P. Cudhea
W. Edmond
L. Evenchik
G. Falk
J. Goodhue
S. Groff
R. Gurwitz
J. Haverty
F. Heart
J. Herman
R. Hinden
D. Hunt
E. Hunter
S. Kent
W. Mann
A. McKenzie
D. McNeill
W. Milliken
M. Nodine
R. Rettberg
R. Reynolds
J. Robinson
E. Rosen
P. Santos
J. Sax
S. Storch
R. Thomas
B. Woznick
Library

END

FILMED

7-83

DTIC