BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

BBN Report No. 3210

November 1975

AD A018660

# DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

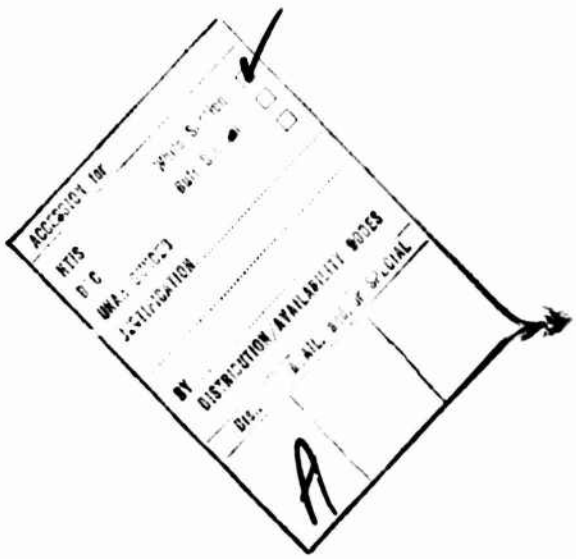
Quarterly Progress Report No. 4
1 August 1975 to 30 October 1975

DDC
RECEIVED
DEC 30 1975
C

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the United States Government.

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2901, Contract No. N00014-75-C-0773.

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.


ACCESSION for
NTIS
DDC
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist.
A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(14) 1. REPORT NUMBER: BBN [redacted]-3210

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

(6) 4. TITLE (and Subtitle): DISTRIBUTED COMPUTATION AND TENEX-[redacted] RELATED ACTIVITIES.

5. TYPE OF REPORT & PERIOD COVERED: Quarterly Progress 8/1/75 - 10/31/75

6. PERFORMING ORG. REPORT NUMBER

(15)

(10) 7. AUTHOR(s): J. Burchfiel, R. Thomas, T. Myer, R. Tomlinson

8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0773, ARPA Order - 2935

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Mass. 02138

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

(11) 12. REPORT DATE: Nov[redacted]75

13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

(12) 39p.

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

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

(9) Quarterly progress rept. no. 4, 1 Aug - 30 Oct 75

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2935.

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

distributed computation
distributed data bases
distributed file system
TENEX security
National Software Works
message processing
CINCPAC interactive test
RSEXEC

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

This report describes continuing refinement of the TENEX RXEXEC distributed file system which supports geography-independent computing on a number of ARPANET TENEX sites. It describes BBN efforts to integrate TENEX into the National Software Works system. It also describes BBN efforts to upgrade existing ARPANET message service to meet NAVY requirements for an interactive message processing test at CINCPAC.

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

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060 100.

TABLE OF CONTENTS


| | | Page |
|---|---|---|
| I. | INTRODUCTION | 1 |
| II. | DISTRIBUTED COMPUTATION | 4 |
| | A. Maintenance of Distributed Data Bases | 4 |
| | B. RSEXEC and Distributed File System | 6 |
| III. | NATIONAL SOFTWARE WORKS | 8 |
| | A. Extending the Encapsulation of TENEX Tools | 8 |
| | B. A Limited Capability File Package | 10 |
| | C. Automatic Startup of NSW | 10 |
| IV. | TENEX RELATED ACTIVITIES | 12 |
| | A. Internet Protocol | 12 |
| | B. Security | 14 |
| | C. TENEX 1.34 | 14 |
| | 1. RCTE | 14 |
| | 2. Measurements on TENEX CD230 Disk Driver | 17 |
| | 3. A Noteworthy Bug-fix | 17 |
| | 4. Improvements to the Network Lineprinter Spooler | 18 |
| | 5. BCPL Compiler | 21 |
| V. | COTCO ACTIVITIES | 22 |
| | A. MAILSYS Development | 23 |
| | 1. Templates | 24 |
| | 2. Sequences | 25 |
| | 3. Other Extensions | 26 |
| | B. Graphic Text Editor - Preliminary Design | 30 |
| | C. Message Storage and Retrieval | 32 |
| | D. System Security | 33 |
| | 1. Top Level MAILSYS | 33 |
| | 2. JSYS Trap Environment | 33 |
| | 3. Access Isolation Module (AIM) | 34 |
| | E. Statistics | 34 |
| | F. LDMX Interface | 34 |
| | G. Mail Forwarding Improvement | 34 |

# I. INTRODUCTION

During the last quarter our "Distributed Computation and TENEX Related Activities" ARPA research has continued to make progress in the areas of:

- distributed computation
- support for the National Software Works
- internetwork protocols
- TENEX extensions and improvements
- message technology

Details of this work are included in the sections of this report that follow.

In the distributed computation area our work on data bases has lead to the development of a new algorithm to coordinate multiple, simultaneous updates to a distributed data base in a way that guarantees both the internal consistency of each distributed data base component and the mutual consistency of the collection of data base components. In addition, the distributed file system supported by RSEXEC has been significantly improved with the addition of the "*" or "wild" designator for file names. The "*" feature has been integrated into the RSEXEC command language and is available to programs that execute in the RSEXEC environment.

Much of our effort this quarter in support of the National Software Works (NSW) Project has been to prepare for the initial

demonstration of the NSW system which is scheduled for mid-November at Gunter AFB. We have continued to work closely with Massachusetts Computer Associates to insure that TENEX functions effectively in the NSW system. Our NSW work is detailed in Section III.

As part of our TENEX related activities, we completed implementation and initial checkout of the basic transmission control program (TCP) for TENEX which implements the internetwork protocol initially specified by Cerf and Kahn. During the checkout phase, debugging sessions conducted with the TENEX TCP and one implemented for a PDP-11 at Stanford by Cerf's group, uncovered several deficiencies in the protocol itself. Revisions to the protocol have corrected these problems.

For some time we have been using a simplified version of the internetwork protocol to support the cross-network lineprinter spooler for our TENEX hosts at BBN. Regular production use of this protocol has uncovered two problems which adversely affect TENEX performance. Both problems, which are described in detail in Section IV.C.4, have been corrected this quarter.

In other TENEX activities related to the network, we have completed implementation of the TELNET remote-controlled transmission and echoing (RCTE) option which will be released with version 1.34 of TENEX. Users who access TENEX through the ARPANET from hosts that also implement RCTE should perceive a significant improvement in the responsiveness of character echoing. In addition, the network traffic required to support remote terminal

access to TENEX should be reduced significantly.

Our work in the COTCO message technology project has resulted in the first release of a new, second generation MAILSYS (Version 2). This new MAILSYS represents a significant improvement over version 1. It provides a more uniform and powerful set of message processing primitives and incorporates a set of system defaults carefully designed to minimize the typing required to accomplish common operations.

In order to support the security requirements of the COTCO project, we have begun the preliminary design for an access isolation module for TENEX. This work is being done in cooperation with a computer security group at MITRE which has had experience with the application of similar techniques to a MULTICS installation now operational in the Pentagon.

## II. DISTRIBUTED COMPUTATION

### A. Maintenance of Distributed Data Bases

Our work on multiple copy, distributed data bases has resulted in the development of an algorithm for maintaining such data bases which we believe satisfies the following properties:

1. Distributed updating.
   Updates to a redundantly maintained data base can be initiated through any of the data base sites.

2. Update Synchronization.
   Races between conflicting, "concurrent" update requests are resolved in a manner that maintains both the internal consistency of each data base copy and mutual consistency of the collection of data base copies.

3. Deadlock prevention.
   The synchronization mechanism that resolves races does not introduce the possibility of so called "deadly embrace" or deadlock situations.

4. Robustness.
   The data base maintenance mechanism can recover from and function effectively in the presence of communications (network) and data base site (host) failures. The algorithm is robust with respect to lost and duplicate messages, the (temporary) inability of data base managing processes to communicate with one another (due to network or host crashes), and the loss of memory (state information) by one or more data base managing processes. In developing the algorithm, we rejected any mechanism that required all data base managing processes to be up and accessible in order for it to function effectively. We sought a mechanism that requires only pairwise interactions among the data base managing processes.

5. Provable correctness.
   It is possible to prove (or at least make a strong plausibility argument for) the correctness of the mechanism.

The following is a brief overview of the algorithm.

The data base copies are assumed to be accessible at each data base site only through data base manager processes (DBMPs). To initiate an update, an application process (AP) may submit an update request to a DBMP. The collection of DBMPs then act cooperatively to perform the requested update and notify the AP of its acceptance or rejection. An AP process is free to resubmit a rejected request for reconsideration by the DBMPs.

The DBMPs determine whether to accept a given update request by voting on it. A request that receives a majority consensus from the DBMPs will be accepted. A single dissenting vote is sufficient to reject a request. The race resolution mechanism may require that a request be rejected in order to maintain the consistency of the data bases.

For example, consider a 3 DBMP system for a data base which includes the variables x(=1) and y(=2). Suppose that two application processes, AP1 and AP2, concurrently request the updates x:=y and y:=x, respectively, by initiating the requests at DBMP1 AND DBMP3. After the two updates are completed, one would expect x and y to be equal, although one could not predict whether their value would be 1 or 2. If both requests were to be accepted, x and y would not be equal. Hence, one of the requests must be rejected in order to maintain the (internal) consistency of the data base. Stated somewhat differently the update request that gets rejected must be refused because it is based on information made obsolete by the request that gets accepted. The AP whose request is rejected is

free to resubmit it. If, when it is resubmitted, the request is based on current information, it can be accepted.

We are preparing a report that describes the algorithm in detail. The report specifies the DBMP voting procedures and includes a "proof" of the correctness of the algorithm.

B. RSEXEC and Distributed File System

During this quarter, the file name syntax for the RSEXEC distributed file system was expanded to allow use of "*" as a "wild" designator. The "*" designator can be used in file names within the distributed file system environment in much the same way as it is used within a single site TENEX environment. The difference, of course, is that within the distributed file system environment "*" has a multi-host scope. For example, the command

```
-DELETE (files) *.*
```

would cause all files in the user's (multi-host) composite directory to be deleted; the command:

```
-NEED (files) A.*  *.B ,
--(in component directory) [ISIC]<JONES>
```

would cause a copy of every file in the user's (multi-host) composite directory with name A and every file with extension B to be moved to the specified file directory.

The "*" feature has been integrated into the RSEXEC command

language so that RSEXEC file system commands behave like the corresponding single TENEX EXEC commands. For example, "*" is the default value for name, extension and version fields in file name arguments for the DIRECTORY and QFD commands. In addition, programs executing within the RSEXEC environment can make use of "*" in file names (via GTJFN, GNJFN, etc.). Thus, it is now possible to run programs such as DELVER (which cleans up file directories by deleting old versions of files) within the distributed file environment.

The latest version of RSEXEC which includes the "*" feature as well as a number of other improvements has been running on the BBNA system since mid-October. After we feel that it has been exercised sufficiently, we will distribute it to all other TENEX sites as the last "official" release of RSEXEC. We expect this will happen early in January.

We presented a paper at the Fifth Symposium on Operating System Principles on the TIP/RSEXEC system titled "An Operational System for Computer Resource Sharing." The paper, which was prepared jointly with the Computer Systems Division at BBN, will be published in the symposium proceedings which will appear as a special issue of the ACM SIGOPS Operating System Review.

## III. NATIONAL SOFTWARE WORKS

The major emphasis of our National Software Works (NSW) project activity for the past quarter was aimed toward a demonstration of initial NSW system capabilities very early in the next quarter. In preparing for the demonstration and pursuing the general development of tool bearing host (TBH) software, we have made significant improvements in our encapsulator for TENEX tools and have designed and implemented an initial version of a program to handle intra-host file operations for TENEX. In addition, we have specified and begun to implement mechanisms which would automatically create the NSW environment in response to a network request. We have also met with project members from Massachusetts Computer Associates in an effort to develope an approach to the long term NSW communication needs.

### A. Extending the Encapsulation of TENEX Tools

As noted in the previous QPR, our approach to utilizing TENEX programs as NSW tools is through the trapping and translation of TENEX system calls into calls meaningful in the NSW system. This NSW encapsulation allows TENEX programs to become NSW tools with very little modification. We have substantially improved the currently running version of the Encapsulator in number of areas.

The Encapsulator now provides simple editing capabilities to the user during the collection of NSW file names on behalf of the tool. To eliminate a cause of lengthy delays throughout a tool session, The Encapsulator has adopted a strategy whereby it waits

until the end of the tool use to deliver any files to the Works Manager. During the course of a tool session, the Encapsualtor maintains those files which need to be delivered when the tool terminates. A primary benefit of this approach occurs because one file will often supercede a previous copy. When this happens, only one version of a particular file need actually be delivered, the others being temporary checkpoints created during the tool session. When this type of file usage is prevalent, as it is with many text editors, the encapsulation file delivery approach achieves substantial savings. These savings are translated to the user in terms of a more responsive tool.

A side effect of implementing this strategy is that we have also had to implement a version number mechanism to allow a user to refer to one of a series of files he has asked to have saved using a common name. The version numbers are in effect only for the duration of the tool session, and can be used to specify a file other than the appropriate default version for input or output. Currently, only the highest version of a particular file is permanently saved in the NSW file system when the tool terminates. However, we plan to modify the Encapsulator to allow the user to explicitely specify which versions he would like to keep.

Also during this quarter, the Encapsulator was modified to adopt to the new interim intra-host communication facility (MSG) adopted in August. Using this facility, and in cooperation with NSW project members from Massachusetts Computer Associates and Stanford

Research Institute, we have integrated the Encapsulator with the Front End and the Works Manager. After much inter-organizational debugging and testing, we have been able to run a TENEX text editor under encapsulation in an actual, but simplified, NSW environment.

B. A Limited Capability File Package

We have designed and implemented a TENEX NSW file package which on command from the Works Manager handles the intra-host file movement and management functions necessary for the initial NSW testing environment. The file package currently supports local file copying and file deletion requests. We have integrated the use of the file package with the Works Manager file manipulation routines using the intra-job MSG facility. Two concepts embodied in the file package program are of special interest. The file package process can operate on lists of files specified in a single request, along with having the capability for reporting partial success. In addition, it has the ability to generate a unique TENEX file name when copying a file, thereby relieving the Works Manager from concern with the syntax governing the TENEX file system.

C. Automatic Startup of NSW

In order to make it easier for a user to start the initially configured NSW system, we have begun to specify and implement modules which provide for the automatic setup of the NSW environment in response to a network request. At the current time, in order to

enter the NSW system, a user must first invoke his local host Telnet, try to connect to a site which maintains the NSW system, log into that site, ask that system to run the NSW, and finally log into the NSW. Our proposal would allow a user to give the "NSW" command to his local host, which in cooperation with software running at an NSW site would bring the user directly to the point of logging into the NSW. We have specified the modifications necessary to existing NSW components to adequately implement the automatic NSW setup. These include having MSG recognize the conditions under which it should terminate to indicate the end of the NSW session, and a Works Manager timeout for monitoring the successful login to NSW. Additionally, we are adapting the previously implemented PCP dispatcher (a piece of code made obsolete by the change in NSW communication protocols) to serve as the background process which listens for NSW requests on a selected ARPANET host socket. The dispatcher will be programmed to respond to an ARPANET Initial Connection Protocol (ICP) sequence to this socket by creating a new job logged in to TENEX which automatically begins execution of the NSW software. We anticipate a completed implementation early in the next quarter.

## IV. TENEX RELATED ACTIVITIES

### A. Internet Protocol

The basic Transmission Control Program [Kahn _ Cerf, International Network Working Group Memorandum No. 39] has been implemented and debugged. Extensive metering and internal statistics gathering facilities have been included. These have proved invaluable in both packet throughput measurements and in finding bottlenecks in the TCP code.

The TENEX TCP has been tested by communicating with the TCP at Stanford University Digital Systems Laboratory. During these sessions several deficiencies in the protocol were discovered. In particular, the protocol for closing a connection (the FIN control function) was inadequate. This issue has been resolved in meetings between Stanford and BBN. The TENEX TCP now uses the revised version of the protocol.

One of the main features of the TCP is its ability to periodically resynchronize connections in order to guarantee that the sequence numbers used in packets will never conflict with any which might be in use in packets or their duplicates which could be reverberating in the network. This has been thoroughly tested and demonstrated to work if each end of the connection independently resynchronizes, and if both ends simultaneously resynchronize.

Several auxiliary programs have been written which use the Internet Protocol. ECHO is a simple program which echos messages sent to it. TTLSRV is a simple TCP Telnet server which allows logging into BBNA from a remote site using the Internet Protocol. TTLUSR is the user companion to TTLSRV -- it is allows terminal communications with a TCP at a remote site which is running TTLSRV.

TCPTST is a test program which opens a TCP connection to itself. One process in the program servers to send Internet letters over the connection which are received by another process. In manual mode these letters are simply lines of text typed on the terminal (very similar to an echo program). In automatic mode the letters are produced by the program. The receiving process checks for errors in sequencing and data in automatic mode.

GATEWAY is a very simple gateway program which runs on BBNB. Operationally, GATEWAY forwards internet packets received on ARPANET link 155 onto link 158 and vice versa. This permits experimenters to view link 155 as "ARPANET A" and link 158 as "ARPANET B" (Internet numbers 12 and 13). Two TCPs may be operated, one in each logical network and messages destined for a host in the other network are sent to the GATEWAY program for forwarding.

Current activity in the TCP is directed towards improving the packet throughput which is currently only approximately 23 packets per second. The largest increase in throughput was achieved by hand coding only about a dozen routines and resulted in a 146 per cent increase. It is expected that improvements in the basic TCP

algorithms will result in a significant increase in the future.

## B. Security

A preliminary study of the feasibility of implementing the Access Isolation Mechanism into TENEX has begun. This is being done in conjunction with a computer security group at MITRE corporation which applied similar techniques to a MULTICS installation now operational in the Pentagon. The meetings to date have resolved most of the major design questions, leaving only a few difficult issues to be resolved.

The current design is oriented towards implementing a multi-level security system. A careful study of all TENEX system calls has been done with the goal of identifying all storage channels. Methods of eliminating these channels are being proposed. In parallel a list of all covert channels is being prepared so that the limits of the eventual system will be catalogued.

## C. TENEX 1.34

### 1. RCTE

The code to implement the RCTE option of the new TELNET protocol for TENEX has been completed. The RCTE option permits a reduction in network traffic by deferring the transmission of characters which will not cause the receiving user program to be activated until a character which will cause the user program to

be activated. A further reduction is achieved by minimizing the flow of echo characters back to the user TELNET program. This is done by having the server instruct the user TELNET to echo the group of characters up through the next wakeup character. By sending this command as the user program is about to read the first character of that group, the echo is guaranteed to follow any response to the preceding group of characters.

Significant problems with the RCTE protocol were encountered. The handling of spontaneous output was specified in a way that made the implementation extremely difficult to do correctly (if, indeed, a correct implementation is possible). The solution here was to completely isolate the control of input transmission and echoing from the characters flowing in the output stream. Synchronization of input and output then occurs directly by virtue of the embedding of control information in the output stream. This approach permits a simplified coding of both the user TELNET and server TELNET.

A second problem was the handling of interrupt characters. The RCTE protocol fails to provide an explicit mechanism for interrupt characters thus necessitating the handling of interrupt characters as program wakeup characters. Since interrupt characters in TENEX are not actually handled as program wakeup characters and, in fact, bypass the terminal input buffer, a special provision had to be made to get the

command sent back to the user TELNET to indicate that the character stream should be echoed beyond the point where the interrupt character was typed. The transmission must be synchronized with the processing of the terminal input buffer so that it will be sent at the proper time. This was achieved by putting a marker in the input buffer at the point where the interrupt character was. This marker is never given to the user's program and must not be counted as an input character. A new counter was installed indicating the number of such markers in the input buffer and the SIBE JSYS was modified to indicate "empty" only if the difference of the total characters in the buffer and the number of markers in the buffer is greater than 0.

A third problem is handling the case where a new program clears the input buffer. Since the buffer may contain various wakeup characters and special markers, when the buffer is cleared, the user TELNET and SERVER may get out of synch. It is infeasible to scan the buffer and send a RCTE command for each such wakeup character or special marker. Instead, a command is sent to the user TELNET meaning "clear your input buffer and reset your counters". This command is implemented by sending "WILL RCTE". This is the reverse case from a normal RCTE (i.e. DO RCTE) and thus cannot be confused with the normal use of the RCTE option. This saves adding a new option.

## 2. Measurements on TENEX CD230 Disk Driver

During this quarter, an effort was undertaken and completed to measure the delays incurred by a disk-transfer task during the various stages of its existence. This effort was motivated by a desire to determine the adequacy of the CD230 disks as a swapping medium for a KA-10/TENEX system. This new facility measures delays due to queuing, arm positioning, rotational latency and actual transfer. The data is segregated by direction of transfer (read/write). It is not possible at this time to provide a definitive interpretation of the results obtained to-date, as further experimentation is necessary and is presently being conducted. The results of this work will be reported at some future time.

## 3. A Noteworthy Bug-fix

For some time, the 1.34 version of TENEX has been plagued by very occasional instances of dreadful scrambling of the various lists of buffers maintained by the network control program. The situation defied explanation for a considerable period of time, though attempts were made to solve the problem by all of BBN's senior systems people. While we understood that we clearly were faced with a low-probability race-condition, it resisted our attempts to analyze it partly because of its infrequency, but mostly due to the absence of usable clues in the post-mortem

dumps we collected. We finally succeeded in locating the problem after adopting the approach of installing successively tighter consistency checks in the NCP, the checks being suggested by inconsistencies observed in prior post-mortem dumps. The error turned out to be a missing instruction in the process-creation code which caused the NCP to receive spurious wakeup signals. If the NCP happened to be waiting for the network-buffer free-list to become available, the result would be chaos.

## 4. Improvements to the Network Lineprinter Spooler

Two problems had shown up after regular production use of the line printer spooler based on the Kahn-Cerf protocol (previously reported). These were, first, the ability to cause the IMP to completely choke off all network traffic to and from a TENEX host, and second, an excessive amount of CPU load running the spooler.

The problem of choking off a TENEX's network interface occurred as follows. The line printer protocol was based on a model of a lossy network, with flow control provided by having a fixed window size representing the amount of un-acknowledged data in the net at any moment. Only that fixed amount of traffic would be sent ahead, and it would be repeated at an infrequent interval after timeouts. In practice, the receiving site (the

line printer) would block its IMP interface when it had no free buffers for line printer information, and this caused overflows in its IMP when further traffic arrived for the line printer. This had been partially solved by reserving one network input buffer for overflow, throwing away any messages received when all other buffers were full. However, if the line printer controller became unresponsive for any reason, it might not be able to discard this overflow traffic. The IMP would mark the host as down, and would return a Host Dead message to the sending TENEX. The spooler in the TENEX would re-send the data on a timeout basis. This timeout took no account of the Host Dead replies. The IMP would then try to reply "Host Dead" to each retransmitted message. However, the IMP system is unable to generate these replies quickly enough to keep ahead of the sending TENEX. Eventually the IMP blocks the TENEX interface for periods of the order of half a minute. This period is of the same order as the spooler's retransmission timeout, so the situation snowballed and the entire network became blocked at the TENEX port. This situation was corrected in a new version of the spooler system by taking proper notice of Host Dead replies from the IMP and by implementing a SLOW or FAST mode for sending the data to the line printer. The system reverts to the SLOW mode on any timeout or Host Dead message, and only uses the FAST mode when data acknowledgements are returning promptly from the line printer. This problem does, however, point out a serious problem in the IMP: it should not be possible to

completely block off traffic to the network as a result of sending traffic to one unresponsive host. This will be largely solved by the IMP's non-blocking host-to-IMP protocol change.

The second problem, excessive CPU usage, resulted from three factors: desire to count pages of paper sent by the TENEX, an unfortunate choice of data format through the network, and the extra overhead of RFNM processing caused by the change described above for properly handling control messages. The accounting of pages required a check of each character for a formfeed. This facility was removed. If it becomes desired again, it should be placed in the minicomputer at the line printer, not in the TENEX. The format of the data in the network messages had been chosen for generality, not efficiency. The text was sent in eight bit bytes as the PDP-11 wanted to see them, rather than as the seven bit bytes which the PDP-10 normally stores them. This conversion load was placed on the TENEX system, rather than on the PDP-11 which can more easily spare the computation time. This has now been changed so that the PDP-11 does this conversion. These changes produced a reduction of more than fifty percent in the computation load on the TENEX by the spooler. The load added by the processing of RFNM messages has not been removed at this time.

## 5. BCPL Compiler

This quarter, we have continued our debugging of the new BCPL compiler, and have released it to the active BCPL users for use and comment. Several bugs were discovered this way, and have been fixed. Currently, the compiler has compiled itself, the current version of BDDT on system A, and the experimental version of MAILSYS. All of these programs appear to be functioning normally.

Also this quarter, we were able to significantly decrease the total amount of free storage needed for the tree, by returning to free storage all tree nodes as they are simplified. This enables the compiler to handle larger programs without running out of storage space.

Initial figures indicate that code generated by the new compiler is 20-25% shorter and runs 15-30% faster than code compiled by the previous version. These figures are based on several large programs which were compiled by the new compiler but loaded with the old BCPL library, so the savings may be somewhat larger than this.

## V. COTCO ACTIVITIES

In the last Quarterly Progress Report, we discussed the evolution of a new MAILSYS Version from the original "straw man" system that was released in April 1975. The major work in the current quarter has been to carry that design effort to completion with the release of MAILSYS Version 2.1 in October, 1975. At this writing, analysis and further refinement of the system is in progress, and plans are complete for the release of Versions 2.2, 2.3, and 2.4 over the next few months.

Another aspect of our efforts has been the preliminary design of new features and capabilities to be integrated into the system during calendar 1976. During the last progress report we briefly described the first of these efforts, namely, a small scale text editing facility intended for integration into the message writing portion of MAILSYS, and intended mainly for printing terminal application.

During the current quarter, we have completed two more such initial design studies. The first concerns an alternative form of the text editor intended specifically to take advantages of the special features of display terminals. The second study attacks the issue of message storage and retrieval and proposes a simple tree structured indexing system to serve in lieu of today's dependence upon the underlying Tenex file system.

During this quarter we have also continued our work on system security and the underlying message delivery system. We have refined the MAILSYS statistics package and reporting program to generate more meaningful output and correct certain errors. Work on the LDMX - TENEX interface has continued, and a basic level of implementation is nearing completion.

## A. MAILSYS Development

During October 1975, we released MAILSYS Version 2.1 - the first release of a new second generation MAILSYS design. That system is now in use and under review and testing by members of the project staff and other interested users within BBN, together with NAVCOSSACT, NAVELEX, and ARPA personnel. Results of this review process have suggested a number of refinements and lead to the creation of a detailed plan for releases 2.2 2.3, and 2.4!

Three basic goals have motivated the development of MAILSYS Version 2. First, as reported in the last QPR, we wished to respond to the many findings that resulted from the human factors analysis of Version 1. As previously reported, this resulted in a more uniform and powerful set of message processing commands, together with a unique system of defaults that minimizes the typing required when common operations are to be performed.

Our second objective in this development was to produce a more compact and efficient implementation. We have speeded up many of the key operations such as the parsing of a message file. We have

also recoded selected portions of the system in the interests of greater modularity and compactness.

MAILSYS Version 1 provided a useful set of basic message processing tools, but omitted numerous features that would extend its power and convenience. Our third goal in developing Version 2 was to overcome certain of these omissions by provision of new features. As we shall discuss later, we intend that this process shall continue as we design subsequent versions of the system.

1. Templates

MAILSYS Version 1 provided a choice of two or three different output formats for listing or printing messages. In Version 2 we have greatly extended the user's control over output format through the mechanism of templates. A template is a user created "map" that specifies what message fields are to be printed in what order, whether or not they are to be labeled, where they are to appear on the page, and what (if any) arbitrary text is to be included for format or labeling purposes. The user is given the power to create, name, store, and use any number of templates of his choice. In addition, the system supplies a number of standard templates for full and summary style output.

Templates are created and named by the user through a series of editing commands specialized to deal with this type of object.

These commands add or remove lines or portions of a line, and provide for displaying the partially completed template.

## 2. Sequences

As a first step toward a general indexing and retrieval capability, we have created a new object known as a message sequence. A message sequence is simply a named group of messages collected together by the user for some particular indexing purpose. A sequence might, for example, contain all messages from some particular correspondent, all messages relevant to a certain technical topic, or all messages falling between two dates. Message sequences are thus a named and stored version of what the user can enter "on the fly" as arguments to commands such as PRINT or LIST. Sequences are created by means of a series of special editing commands (discussed below) and named at the time of creation. A named sequence can be supplied as argument to any of the message output commands instead of or in addition to the set of messages specified by the user when the command is given. Note that a message sequence is not actually a physical collection of messages, but rather a collection of references into the currently active message file. Thus, there can be any number of concurrent sequences, reflecting the possibly numerous subsets and partitions into which a user might wish to organize a message file.

As with templates, the user is provided with a set of commands for constructing and editing sequences. These include the ability to add messages to a sequence, to remove messages, to logically intersect one sequence with another, to apply a filter to a sequence, and to sort the messages in a sequence based on any of several different possible sort keys derived from the message header field. These latter include date, author, subject, etc. Note that one can physically sort a message file by creating a sequence that references all messages in the file, sorting the sequence according to the criteria desired, and then transfering all messages, in the sorted order to some output file.

### 3. Other Extensions

We have also extended and improved the handling of two other types of objects: filters, and the user profile. The commands available for editing, applying, and displaying filters have been considerably improved. The entire user profile has been made accessible to the user as an object. The user can now examine and change the contents of his profile in the same manner as other objects such as templates or sequences. He can specify that created objects such as templates or filters be added to or removed from his permanent profile. He can alter the various switch settings and other objects included in the profile and examine their current values.

In order to provide uniformity within the system, any message currently under creation by the user is itself regarded as an object, known as the "draft message". The act of creating, modifying, and displaying the current state of this draft message is accomplished by a set of editing commands equivalent to those that apply to the other objects.

This state of affairs is shown in Figure one which depicts the current organization of the major processing tools in MAILSYS. As shown, the major active objects are the draft message, filters, message sequences, and templates. Each of these objects is subject to processing by a collection of edit commands. In addition, there is a collection of "top level" object manipulation commands that perform functions such as displaying an object, copying from one object into another, deleting an object, and the like. Output from the currently active message file is accomplished by a uniform collection of transcription commands which accept message sequences and templates as their major arguments. Message creation is accomplished either by editing the draft message, or by prompted input sequences such as SNDMSG and DD173.

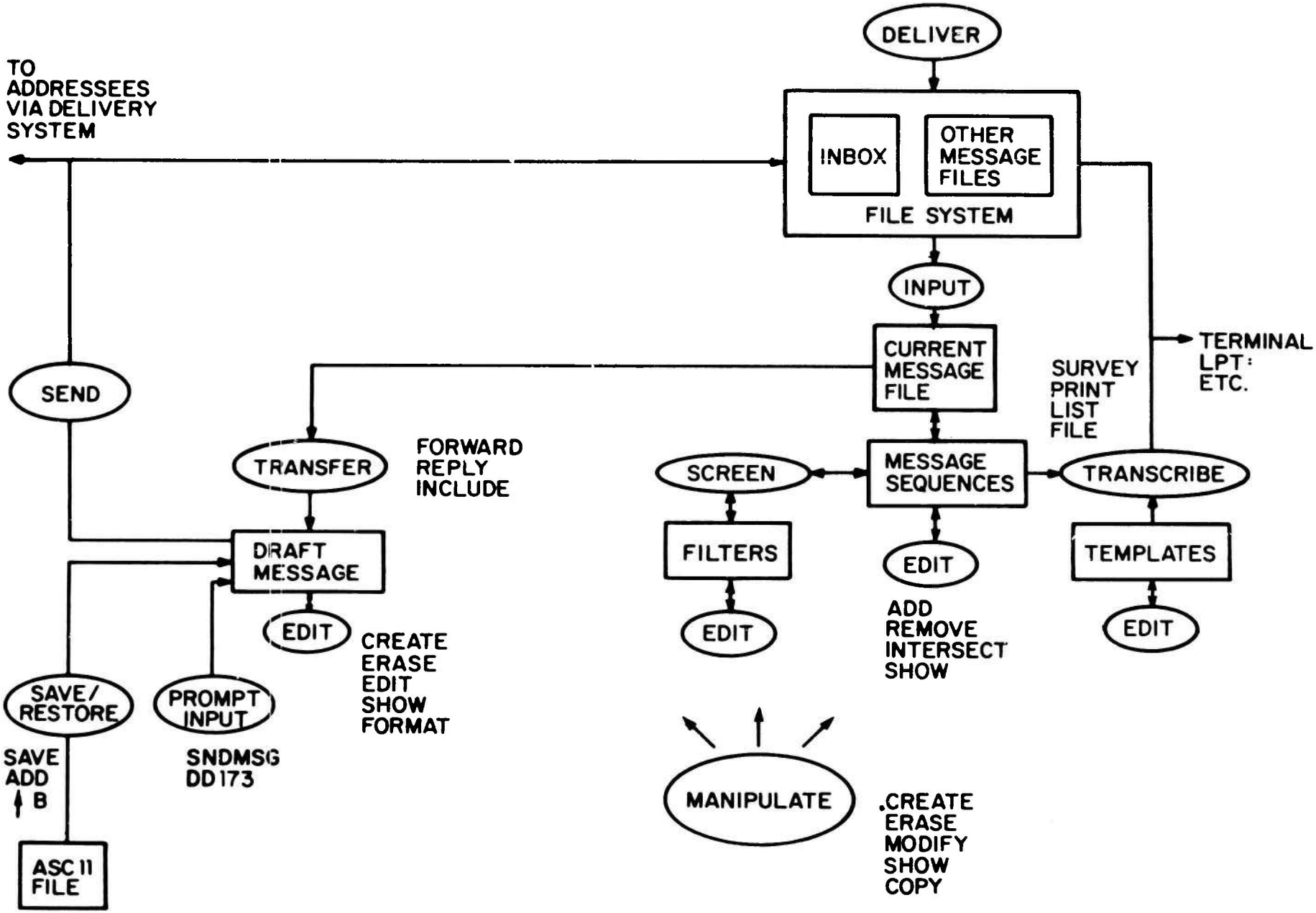

DELIVER
TO ADDRESSEES VIA DELIVERY SYSTEM
INBOX
OTHER MESSAGE FILES
FILE SYSTEM
INPUT
CURRENT MESSAGE FILE
TERMINAL LPT: ETC.
SURVEY PRINT LIST FILE
SEND
TRANSFER
FORWARD REPLY INCLUDE
SCREEN
MESSAGE SEQUENCES
TRANSCRIBE
DRAFT MESSAGE
FILTERS
EDIT
TEMPLATES
EDIT
EDIT
CREATE ERASE EDIT SHOW FORMAT
EDIT
ADD REMOVE INTERSECT SHOW
SAVE/ RESTORE
PROMPT INPUT
SAVE ADD B
SNDMSG DD173
MANIPULATE
.CREATE ERASE MODIFY SHOW COPY
ASCII FILE

FIGURE 1

# CONTROL STRUCTURE OF MAILSYS

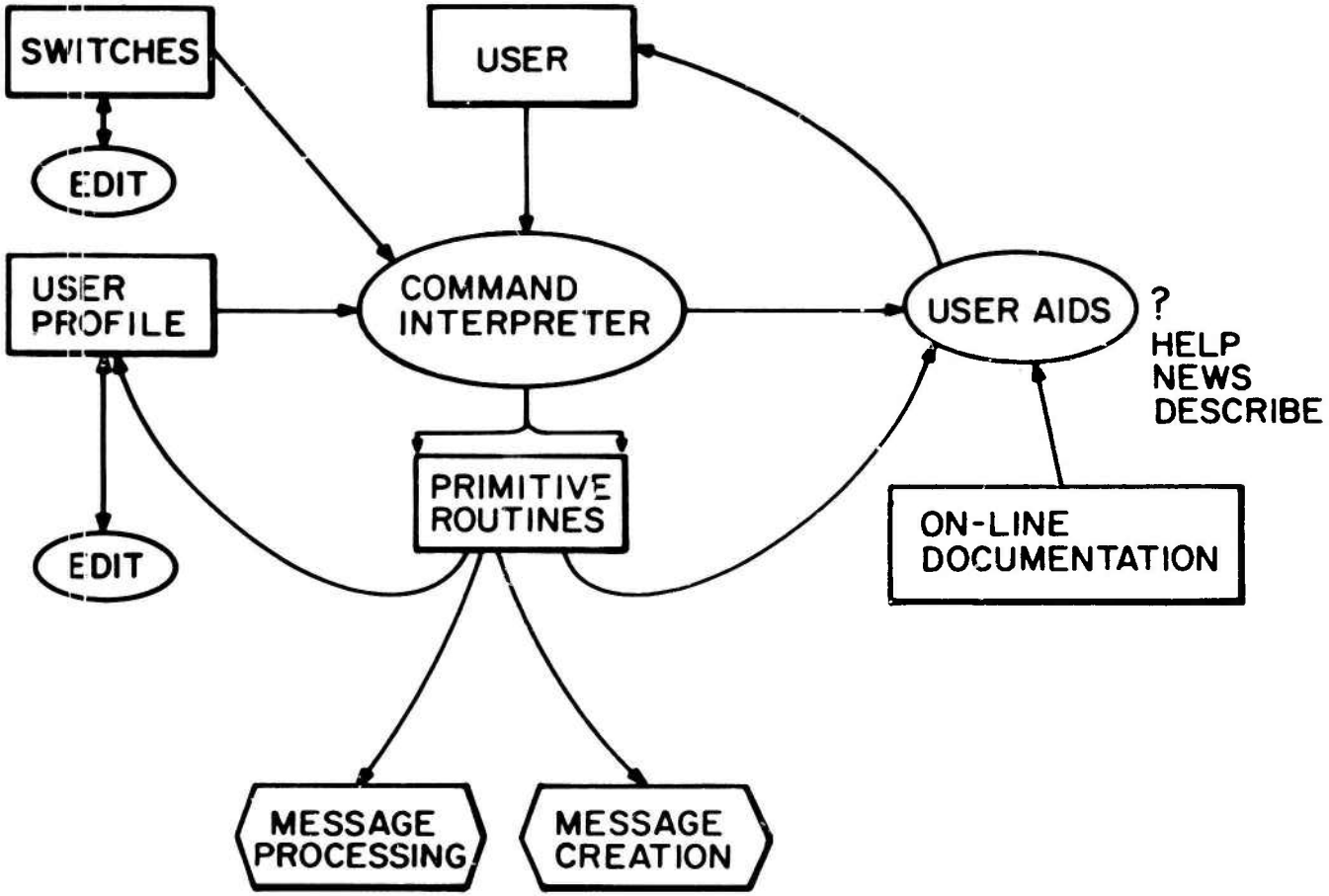



FIGURE 2

Figure two shows the major processing functions and objects in the control structure of MAILSYS. As indicated, there is a collection of option and mode switches, and the user profile, each of which is subject to creation and manipulation by means of a series of edit functions.

B. Graphic Text Editor - Preliminary Design

Our last progress report described the design of a modest text editing capability to be integrated into MAILSYS. As we discussed, the desire for such an editor was motivated by concern over the present non-integrated text editors that are available in MAILSYS. We feel that the naive user will have trouble mastering the "foreign" command language of these outside editors, and will be further troubled by the mechanics and concept of transitioning to and from a "lower fork".

The initial text editor design assumed a printing terminal, since this represents the lowest common denominator of terminals available over the ARPANET and since one of our mandates is to provide network-wide message service.

It also appears, however, that a growing number of display terminals are available on the ARPANET; and in addition, participants in the Navy Interactive Test will almost certainly make use of this kind of terminal. For that reason, we have undertaken an extension to the editor design that takes advantage of the unique properties of CRT display terminals. These includes the visual

feedback that can be obtained through a moving cursor that can be superimposed upon the displayed image of the text itself; the ability to carry out immediately and display to the user the impact of operations such as insertions or deletions; the ability to perform automatic incremental format control operations such as left and right justification; and the ability to indicate position on a page of text by pointing (or otherwise moving the cursor) rather than by a typed-in command format.

In order to provide for such a display-oriented editor, we have laid out a concise but complete set of commands and command actions. Our view is that this graphic interface to the editing functions would serve as an optional alternative to the more conventional printing oriented editor. Both versions would be available under switch control; those possessing display terminals would most likely wish to switch in the display oriented version.

Naturally, the display techniques that would be so beneficial to text editing would also have a positive impact on the general command structure of MAILSYS. Our long range view is that two complete human interface designs, one oriented toward printing terminals, and one oriented toward display terminals should be available under switch control. We have picked text editing as a first step toward the general display oriented interface, since this function (we suspect) is most critically affected by the differences between the two types of interface.

C. Message Storage and Retrieval

In a third preliminary design effort, we have laid out a tree structured indexing system that would provide a considerably more powerful storage and retrieval capability than is presently available, while at the same time, shielding the naive user from the need for direct manipulation of the TENEX system file structure.

In this scheme, underlying manipulation of physical files would be performed by the index system in a manner invisible to the user. The design proposes a scheme for file management that would also accomplish orderly archiving of old messages on a "flow through" basis.

The design provides for a smooth transition from the present notions of message sequences into the notion of a hierarchical index. The total index takes the form of a tree with groups of messages (or more accurately, groups of pointers to messages in the database below) forming the leaves of the tree. These groups are directly equivalent to today's message sequences, and the user would have at his disposal the same tools for editing, displaying, and sorting the contents of a message group. Whole message groups are addressed by compound names that specify a path through the tree required to access that particular leaf.

Integration with the present message delivery system is achieved by having all new messages added to a permanent leaf of the tree entitled "current". Another permanent category entitled "all"

provides a master sequence numbering facility.

D. System Security

In preparation for the CINCPAC Interactive Test, we have worked closely with MITRE staff members on the problem of security. Work is proceeding on three types of security systems.

1. Top level MAILSYS.

In this system, a class of users would be set up who would have access to top-level MAILSYS only, and not to the full programming and editing capability of TENEX. This requires that some editing and file handling now done by TENEX must be inserted into MAILSYS. We have determined a set of TENEX commands that are candidates for inclusion in top-level MAILSYS and are proceeding to implement them.

2. JSYS Trap Environment.

The JSYS trap mechanism for the TENEX operating system is being used to implement a controlled environment for MAILSYS. Commands given by the MAILSYS user will be automatically trapped and analyzed to see whether the commands are permitted by the User's security status.

3. Access Isolation Module (AIM)

As noted in Section IV.B, we have worked out the basis for the development of an Access Isolation Module for a secure TENEX system. The purpose of this module is to insure that TENEX code is completely isolated from access by other users at the same or other TENEX installations.

E. Statistics

The software package for gathering statistics on the operation of MAILSYS has been completed during this quarter, and the programs for collecting and reporting statistics on system performance have been rewritten to correct errors.

F. LDMX Interface

We are continuing our collaboration with NAVCOSSACT to design details of the protocols of the proposed LDMX TENEX interface. The TENEX portion of these protocols has been 80 per cert designed and 50 per cent implemented.

G. Mail Forwarding Improvement

MAILER, the background process which sends mail to protected mailboxes and to remote sites, was modified to speed delivery of messages which go through an intermediate forwarding site. This change also allows better acknowledgment in the case of mailing

failures. The earlier MAILER sent messages to the addressed site and allowed the addressed site to take responsibility for forwarding them. For example, a user of system BBNA might send a message to SMITH at BBN. If SMITH's mailbox was actually at BBND, MAILER would send the message to BBN and then BBN's mailer would send it to BBND. This resulted in a delay in delivery, and also an extra header line in the delivered message reporting the intermediate transmission. Similarly, in the same situation, if the mail was sent to just "SMITH" rather than "SMITH@BBN", the mail would be sent to SMITH at the local site before being forwarded to BBND. MAILER has now been modified to check the forwarding data base before sending mail locally, thus preventing the extra delivery if the mailbox is found to be on another system. MAILER has also been modified to parse the FTP reply which warns of forwarding, and to use the information in that reply to send the message directly to the correct destination. A side effect of these changes is that the originating site has control of the message right up to the final delivery, so that it can generate a correct failure response if the mailbox turns out to be nonexistent after forwarding (an erroneous, but possible, situation).