AD/A-006 735

DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

Bolt Beranek and Newman,
Incorporated

Prepared for:

Advanced Research Projects Agency

February 1975

BBN Report No. 3012 February 1975

DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

Quarterly Progress Report No. 1
1 November 1974 to 31 January 1975

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 Advanced Research Projects Agency or the U.S. Government.

This research was supported by the Advanced Research Projects Agency under ARPA Order No. 2901.

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

TABLE OF CONTENTS

		Page
I.	INTRODUCTION	1
II.	DISTRIBUTED COMPUTATION	3
	A. TIP Access Control and Accounting System	3
	B. RSEXEC Program Execution Environment	5
	C. Management of Distributed Data Bases	6
III.	TENEX RELATED ACTIVITIES	8
	A. Protocol Development	8
	B. Security	16
	C. Distribution of TENEX Version 1.33	20
	D. BCPL	25
IV.	COTCO ACTIVITIES	27
	A. Background	27
	B. Project Activities	31
	C. MAILSYS	33
	D. Documentation	34
	E. MAILER and FTPSRV	35
	F. TENEX Capabilities	35

I. Introduction

During this quarter most of our work in distributed computing was directed toward bringing the TIP login and accounting system into operational status. As a result, work on the program-level distributed file system and the coupled message service projects progressed more slowly than planned. However, a prototype implementation of a program execution environment that includes the RSEXEC distributed file system has been completed. In addition, we have completed and documented a method for maintaining a (non-trivial) type of distributed, multiple copy data base. This method is to be used both for maintaining multiple copies of the Network User Data Base as required by the TIP login system and for maintaining redundant message files for users of the coupled message service.

During this quarter we also completed a TENEX implementation of the Internetwork Transmission Control Program specified by Kahn and Cerf in the May 1974 IEEE Transactions on Communications. This program will support planned Internet expriments between BBN-TENEX, a PDP-11 at Stanford University, and a PDP-9 at University College London.

The January distribution of TENEX System 1.33 includes a novel resource allocation facility called the "pie-slice scheduler" which permits the facilities administration to slice a TENEX system into guaranteed portions of CPU service for each group of subscribers. This facility will make it possible to plan reasonable allocation of

scarce TENEX resources throughout the ARPANET to various ARPA research projects.

II. Distributed Computation

A. TIP Access Control and Accounting System.

At the request of the ARPA office and together with the Computer Systems Division at BBN we have developed a login and accounting system for ARPANET TIPs (see BBN reports 2869 and 2976). Prior to development of this system, anyone with a terminal and data set who knew the telephone number for a TIP dial-up port had free and uncontrolled access to the ARPANET. TIP login corrects this situation by requiring a TIP user to establish his authorization to use the network by supplying a valid network user name and password.

TIP login and accounting was implemented by building upon the existing TIPSER-RSEXEC system. TIPSER-RSEXEC makes it possible for TIPs (as well as other mini-hosts) to support what are usually thought of as "large host" functions by providing a means for sharing the resources of ARPANET TENEX hosts (see BBN reports 2544 and 2607). In order to meet reliability requirements and to distribute the load among participating TENEX hosts, the TIPSER-RSEXEC is implemented as a distributed, multi-computer system. The login system consists of three distinct, but related, components: multiple network login server processes (TIPSER-RSEXECs); multiple data collection server processes (ACTSERs); and software for reducing accounting data to produce regular (monthl) summaries.

Whenever a user activates a TIP port, the TIP uses a broadcast initial connection mechanism (see BBN report 2507) to select one of the TIPSER-RSEXEC processes to authenticate the user. If the user successfully supplies a valid network-user name and password to the TIPSER-RSEXEC, he is granted continued access to the TIP, the network, and the standard TIPSER-RSEXEC functions. In addition, the TIPSER-RSEXEC transmits the user's unique Network ID code to the TIP (for accounting purposes) and makes a user "login" entry into an "incremental" TIP accounting data file. In order to use a network service host after logging into the TIP, a user must disconnect from the TIPSER-RSEXEC, instruct the TIP to connect to the target host, and then login to the target host.

After the TIP receives the user's unique ID code from the TIPSER-RSEXEC, it activates "connect time" and (outgoing) message counters to accumulate usage data for the user's session. These counters remain active until the user terminates the TIP session. Periodically the TIP executes an "accounting checkpoint" procedure whereby it transmits usage data, accumulated since the last checkpoint for its active users, to an ACTSER process (selected much like the TIP selected a TIPSER-RSEXEC). The ACTSER process stores the checkpoint data in an incremental TIP accounting file for later processing

The collection of incremental accounting files maintained by the ACTSER processes is a large, distributed and segmented data base. The reduction of data in that distributed data base to

produce periodic accounting summaries is accomplished by software which executes within the environment provided by the RSEXEC distributed file syste

At present, TIP access cont ol and accounting is acnieved at the expense of placing an additional access control check between TIP users and the network service sites they plan to use: a user must first login to the TIP and then to the target host. The next planned step in the evolution of the TIPSER-RSEXEC system is to eliminate the second login by extending the effect of the first (TIP) login to include target host login.

3. RSEXEC Program Execution Environment

The prototype version of the RSEXEC program execution environment makes the features of the RSEXEC distributed file system (see BBN reports 2404, 2607, and 2670) available to executing programs. When run within this environment, application programs, such as text e.itors and compilers, can uniformly access files, both local and remote, without regard to the actual network location of the files.

The TENEX JSYS trap mechanism (see BBN report 2721) is used by RSEXEC to provide the distributed file environment to executing programs. RSEXEC arranges to intercept certain operating system calls made by application programs running under its control. For example, certain file operations initiated by application programs are intercepted before the local TENEX monitor acts on them.

Operations that can be handled locally are passed directly to the local TENEX monitor by RSEXEC. Whenever a file operation is initiated that requires access to a remote file, RSEXEC transmits a request across the network to a cooperating RSEXEC server process at the proper host that causes it to execute the operation on behalf of the application program.

The prototype implementation is incomplete in the sense that not all file operations that an application program might initiate are handled. However, we have successfully run a number of standard TENEX application programs, including TECO, RUNOFF, MACRO, LOADER, READMAIL, and MAILSYS, as well as some simple user programs in the prototype execution environment. For example, within this environment we have been able to edit, assemble, load and run programs for which the various source files and load module files were distributed among various hosts in the network without explicitly transferring files from machine to machine.

C. Management of Distributed Data Bases.

We have developed a method for maintaining multiple, distributed copies of a data base in the presence of distributed data base updating in a manner that guarantees the mutual consistency of all copies of the data base. This work was motivated by the reliability & efficiency requirements of the TIP login system which dictated that the Network User Data Base be maintained in this manner. However, we believe that the method has

applicability beyond the TIP login system. For example, the method is directly applicable to the task of maintaining multiple, redundant copies of users' message files as required by the coupled message service.

The method is completely distributed in the sense that it requires no centralized control nor does it require that all copies of the data base be locked simultaneously in order to accomplish the updates. The method (which is described in detail in RFC #677) consists of two parts: a reliable, data independent, update transmission and distribution mechanism which guarantees that all data base updates reach all data base sites; and, a data dependent update procedure which is activated at data base sites when update commands arrive. The update procedure makes use of a "time stamping" scheme which enables data base sites to regenerate a sufficient portion of the time sequence of update events to determine how to consistently incorporate a particular update command into their copies.

III. TENEX Related Activities

A. Protocol Development

1. PDP-11 Internet Prototype

We are currently engaged in network protocol research which is investigating the protocol proposed by Kahn and Cerf (A Protocol for Internetwork Communication, IEEE Transactions on Communications, May 1974) for host-to-host communication between hosts on the same or different networks. Our first experiment in this area (see BBN Report 2670) was a very simplified subset of this protocol which provides reliable transmission of listings from four BBN-TENEX systems across the ARPANET to a PDP-11 which drives the line printer. This protocol is remarkably robust: listings are eventually completed correctly in spite of a temporary service interruption on TENEX, on the PDP-11, or both.

The use of this program brought to light a serious flaw in the current ARPANET host-IMP protocol. The PDP-11 input buffers are filled by network messages much faster than the lineprinter driver (900 lines/minute) can empty them, so the PDP-11 becomes quite unresponsive in taking messages from the IMP once its buffers fill up. The source IMP attached to the TENEX system which is transmitting the listing refuses to allow any more traffic to enter the network until previous messages are successfully delivered to the destination PDP-11. This refusal to accept traffic is done by

locking up the interface from TENEX to its IMP, so all outgoing traffic to other hosts and TIPs only proceeds in small spurts at intervals of tens of seconds. This level of service is intolerable to TIP users of that TENEX system.

The buffer overflow problem occurs in the PDP-11 because the Internet protocol only performs flow control for buffers in the user space which have been allocated to Internet connections. At a lower level (the IMP input driver of the operating system) are a set of receive buffers shared among all incoming network applications: these buffers are not under control of the Internet connection flow control. The observed problem was due to multiple simultaneous attempts (by different TENEX systems) to establish connections to the PDP-11: the requests overflowed the receive buffers.

The cure to the problem of the blocked source host interface is a modification to the host-IMP protocol to permit blocking outgoing traffic to some unresponsive destination with no interference to traffic to other destinations. This can be done by the IMP's accepting all traffic and reporting that messsages to the unresponsive destination were not successfully transmitted. Each time another message can be accepted for the unresponsive destination, the IMP can give the host an explicit go-ahead.

Until the general solution described above can be implemented, we are forced to the temporary expedient of making the PDP-11 always responsive by having its IMP input routine accept and throw away messages when its input buffers are full, resulting in hundreds of "lost" messages per day between TENEX and the PDP-11. The prototype Internet protocol is so robust that these lost messages are always detected and retransmitted: the listings are perfect in spite of the "lossy channel" which now exists between TENEX and the PDP-11.

2. TENEX Internet Experiment

During this quarter, the specification of the transmission control protocol (TCP) for the internetwork experiment and the coding of a TCP for TENEX were substantially completed. As the quarter ended, initial checkout of the code was underway and had demonstrated delivery of data in several test cases.

The TCP implementation for TENEX is being done in user mode utilizing the JSYS trap mechanism (see BBN Report 2721) to augment the user's virtual machine to include the TCP primitives. A separate job holds the main body of the TCP with communication via shared memory and interrupts with the JSYS trap interface program running in the user's job. The TCP uses the network raw message facility to transmit and receive messages on the ARPA net.

The TCP is being coded in BCPL with the exception of the data movers which are coded in machine language to provide additional speed. It is expected that the code will be transferable to the

PDP-11 without major re-write. The principal obstacles to such a move are the need for double precision arithmetic in the sequence number manipulations on the PDP-11 since these are 32-bit quantities, and a different user interface.

3. TELNET RCTE

TELNET RCTE is a protocol which permits more efficient utilization of the ARPA network for terminal IO by allowing more characters to be packed into each message and eliminating most echo characters from being transmitted over the network. The initial implementation of TELNET RCTE (Remote Controlled Transmission and Echoing) in the TENEX TELNET user and server was completed during this quarter.

This implementation uncovered two problems in the specification:

(a) The sender must transfer its bufferload whenever its input buffer overflows: this means that the receiver must be prepared to receive data even before it turns the line around to request input. This complication was not explicitly spelled out in the specification.

(b) In order to provide interrupt characters which take effect immediately even when there has been typeahead, the sender must send its buffer whenever a break character from the previously

specified break set is typed in. This must be done even if the receiver has not yet turned the line around to request input. Of course, the receiver must also declare any such interrupt characters to be break characters as well. This complication was not explicitly spelled out in the specification.

a. RCTE Implementation

In order to make the most efficient use of local echoing of most terminal input under direction of the remote host, a few changes in the wakeup and echoing logic of TENEX were necessary. A new echo mode, viz., echo non-wakeup characters and do not echo wakeup characters, was defined; and the assignments of four control characters to wakeup sets were changed. The new echo mode has been implemented, but the change in wakeup assignments has not yet been made.

The first attempt at an implementation of RCTE in the User and Server TELNETs on TENEX appeared to work fairly well. Unfortunately, it was discovered that it had a major deficiency relating to a point not explicitly addressed in the specification. When the User TELNET was blocked on input, awaiting a break reset from the foreign host, there is no way (in the protocol as defined) to force transmission of an interrupt character to get the attention of the remote processor, analogously to (and literally including) TENEX's control-C.

To solve this problem it was necessary to implement a much more complicated design in which the User TELNET must process input from its terminal even when break-reset-blocked, and transmit when a RCTE break character is input, even if blocked. If blocked, it must also buffer the characters for later echoing. Similarly, the Server TELNET must be prepared to receive input at all times, not just after sending a break reset. This scheme, asynchronous on several levels, is complex, but appears to be the only way to solve the problem of being able to send an interrupt character at all times.

The "second-generation" implementation described above is complete but untested for the User TELNET, and not yet complete for the Server. As an adjunct to the RCTE implementation, a survey of TENEX software has been begun toward the goal of improving the efficiency of this software relative to RCTE operation (using the smallest necessary wakeup sets to minimize packets transmitted). Included is a survey directed toward possible standardization of interactive (control-character) editing conventions.

b. Documentation

The documentation of User TELNET via the User Guide and the program's own Help file was expanded and updated. In conjunction with this effort, a "describe" command feature, similar to that in RSEXEC, was added to User TELNET, to make it

easier for the user to learn the use of the program and to keep up with new features.

c. Other Changes

To improve efficiency and reduce security problems, the logic of typescript files produced by TELNET has been changed. The program now assigns no typescript file unless specifically commanded to do so. Also, when the default temporary typescript file is called for, it is created in the user's LOGIN (not connected) directory, so that it will always be expunged at LOGOUT.

A new command "take.input.stream.from.file" has been added to allow "canned" input from a file (rather than from the controlling terminal) to be fed to a given remote connection.

A new command "auto.switch.to.active.connection" has been added to invoke a state in which several remote connections can be monitored for (presumably occasional) activity. In this state, if a current connection is inactive for a set time, TELNET hunts for another connection which is active; if one is found, TELNET switches to that connection.

4. PCP Modules for National Software Works

Toward the end of this quarter, we began our active participation in the National Software Works (NSW) project. Since

the NSW had been an active effort elsewhere for a number of months prior to our participation, our initial efforts involved becoming familiar with the goals and documented work of the NSW to date, and attempting to clarify our role in the project as a whole. As a result of this inspection, we have determined that BBN's role shall include the design and implementation of the strategies and programs necessary to make BBN TENEX a cool bearing host to the NSW. These tasks include building modules to handle the Procedure Call Protocol (PCP) initial connection and process dispatching, the NSW NVT package, and the modified NSW file package, as well as attempting to make both "old" and "new" tools run correctly in the NSW environment. Our commitment is based on the assumption that SRI will provide us with a working TENEX PCP implementation to support the necessary NSW packages.

Another result of our initial evaluation has been our generation of a set of comments regarding the effectiveness and understandability of the SRI NSW protocol documentation. We are now in the process of distributing these observations to the NSW community.

B. Security

1. CRJOB

The Create Job system call is an important addition to TENEX for two security-related purposes: First, correcting errors in access checks made by previous mechanisms in the system, and second, moving some service processes outside the security kernel of TENEX. A third purpose is related to the Distributed Computation effort, the so-called "double login" problem.

The CRJOB JSYS allows one process (typically, but not necessarily, a privileged system background service) to create a wholly separate job at the request of a user. The created job will have no privileges beyond that of the requesting user. Accounting for services consumed by the created job will be done by the existing accounting mechanisms. Since the created job is "logged in" as the requesting user, the charges will be properly credited.

The present state of implementation of CRJOB allows the system background tasks, the "autojobs", to be logged in as unprivileged users for the cases which do not require special privileges. An example of this is the GRIPEWATCH function (the reporter of RSEXEC-received gripes) which is provided by BBN-TENEX to the Network Control Center. Previously, this job ran as a needlessly privileged process, adding code to the TENEX security kernel. The use of CRJOB in this way required addition of an extra entry into the TENEX EXEC.

Further work will be required to accomplish the same goal for the per-user incarnations of the File Transfer Protocol server, and for the double-login task of the TIP and RSEXEC.

2. The SECURE terminal program.

One possible breach of security which has long been recognized is the unauthorized use of a terminal which is left logged in and unattended by a user. This is especially a problem if the user is a system operator or other privileged person. As a stopgap measure, a short program called "SECURE" was written to lock up the terminal while it is left unattended.

This program requests both the user's password and an additional KEY. It then intercepts all typein and interrupt characters from the terminal until the KEY is entered again. The KEY and password are not echoed. This protects the user's job from unauthorized tampering, thereby preventing misuse of the user's privileges.

Of course, if the terminal is connected through a TIP, the TIP command language is still active, and the terminal itself can be taken away, but the privileged job cannot.

3. ULIST

The operations utility program ULIST (User list) has always had the capability of producing a user list with or without the users' passwords. This program is regularly used to list user names with their disk quotas, directory numbers, and other per-user parameters. Since the password listing function has rarely been used for legitimate purposes, and has always been used by malicious users after a security breach has occurred, this function has been removed from the ULIST program.

4. Account Validation in LOGIN

As originally planned, the user account verification code has been moved from user code (the EXEC) into the monitor. This change will help provide a higher level of system security by preventing unauthorized access to accounts. The VACCT (verify account) and GDACC (get default account) JSYS's have been created to facilitate this change. Several other JSYS's were added to help make the user interface to the accounting system somewhat cleaner.

5. OPRFN - The Operator Function JSYS

A possible source of unreliability and of security violations has been reduced by the addition of the OPRFN system call. In the past, a number of regular operational functions have been performed by the operators through commands to the TENEX MINI-EXEC and Monitor DDT. When these mechanisms are used, any erroneous typing can have

potentially disastrous effects, since they are the mechanisms originally provided for system debugging and repair. Examples of the operational functions performed in this way are: varying the memory size available to the system, "recycling" the network control program to clear it when needed, and switching from attended to unattended operation.

These and other functions have been made available through the privileged system call "OPRFN", which is now used by twenty new EXEC language commands. This reduces the need for the operators to use Monitor DDT and the MINI-EXEC. We may in fact remove this capability from the operators entirely in the future.

6. GFACC Access Check

In cooperation with Xerox PARC, the GFACC system call (JSYS) has been installed. This allows user programs to determine access to files and directories and is used by server programs to determine the access of an arbitrary user to a specified file or directory.

7. Change Password

TENEX 1.34 and EXEC 1.53 support the "CHANGE PASSWORD" command for users. In order to guard against typing errors during inputting of the new password, the EXEC was subsequently modified to require the new password to be typed in twice. Both copies must agree for the command to be successful. This method was chosen in preference to echoing the new password on the user's terminal due to a belief

that passwords should never appear (even on operator terminals) in printed form.

C. Distribution of TENEX Version 1.33

During this quarter, version 1.33 of Tenex was distributed. It included the following enhancements:

1. Pie-slice scheduler

The pie-slice scheduler provides a mechanism whereby Tenex jobs are mapped into service groups ("pie-slice" groups), each group being guaranteed a minimum instantaneous service-level determined by system administrators. A job's group is a function of its current account designator; the specific mapping is described to the system by the system administrators.

In addition to regulation of instantaneous service level, the system also provides for long-term regulation of group cost-effectiveness by assigning an administratively-determined portion (the "K" factor -- 0 <= K <= 1) of unclaimed service guarantee (guarantees made to groups for which there are no logged-in jobs) or "windfall" to the least cost-effective represented group. Setting this factor to 1 provides optimal equalization of long-term group cost-effectiveness.

The pie-slice scheduler is currently in operation at BBN on all

four systems and has been demonstrated to meet its primary design goals, including that of performing its function at virtually no increase in scheduling overhead as compared to the non-pie-slicing version of the Tenex scheduler. It is still being refined, however, as its behavior under varying conditions becomes better understood. JSYS's to obtain information about the pie-slice system were also added. A brief description of each of these JSYS's follows: CGRP (available to wheels and operators only) allows a user to change to pie-slice group while continuing to charge the same account. ATGRP will convert an account to the corresponding pie-slice group. This is used in several places by the monitor. GPSGN returns the pie-slice group that a given job is running in at the current time. GACTJ does the same except that it returns the account of the job. CACCT was also modified. For privileged users, the account can be changed without the pie-slice group also being changed. This was necessary to keep the printer job operating at a reasonable level of service. It is also possible to do a CACCT without it being logged on the logging teletype. This will help avoid congestion caused by the printer job constantly changing accounts.

2. Multi-pack disk swapping

Multi-pack disk swapping reorganizes the disk swap area, dividing the space equally among the center cylinders of each of the available disk drives, as opposed to allocating the space in a contiguous area on a single drive. The central location of the swap areas minimizes average arm excursion, and servicing swapping requests with many devices makes possible simultaneous seek operations and, with certain controllers e.g. 3330 equivalent, simultaneous rotational positioning. Prior to installing this feature, BBN's System C was swap-limited in 256K; we were unable to reduce page-wait lost time to less than 10% under heavy load. With the advent of multi-pack swapping, the system became cpu-limited, even in 192K. Measurement of the time required to read a page while swapping on 5 Calcomp CD230 drives averages approximately 27 ms. under moderate load; this is queuing delay plus actual service time. This compares favorably with 22 ms. for the Bryant drum. We will be performing more such measurements, in varying configurations, in the near future.

3. Non-resident lock conflict strategy

Non-resident locks, such as directory locks, present a problem in TENEX because a process failing to lock such a lock cannot provide the Scheduler with explicit wake-up criteria, such as a test of the state of the lock itself, because, in general, references to non-resident storage are not permissible in the scheduler context

(job and process-private storage is not even addressable, except under special circumstances). In the past, this difficulty was met by setting up a short-dismiss followed by lock-retry loop. The problem with this open-loop approach is the considerable overhead involved in a process wakeup, usually including at least several pager traps and swaps. In fact, lockups have occasionally been observed due to many processes attempting to simultaneously seize a single (popular) non-resident lock, and consuming such inordinate amounts of processor time in the dismiss-wakeup-retry sequence that the process possessing the lock was unable to obtain sufficient service to complete the sensitive routine and release the lock in a reasonable amount of time. The new strategy simply involves the use of a single resident flag which is set by any process failing to lock any non-resident lock (the "set" state indicates that at least one conflict remains unresolved). The failing process then dismisses itself, telling the scheduler to wake it when the flag is cleared (the flag is testable by the scheduler because it is resident, though the lock is not). Any process unlocking a non-resident lock on which a conflict has occurred (this is detectable from the value of the lock itself) clears the flag. This simple algorithm eliminates the vast majority of unnecessary wakeups, failing only when conflicts involve more than two processes, either on the same or different locks.

4. New balance set holding strategy

It has been observed that certain processes, notably the TENEX Network Control Program, and processes running tapes, experience poor service and tend to degrade total system performance. This is because these processes tend to dismiss themselves frequently for relatively short intervals. In the past, a dismissed process was removed from the "balance-set" (the set of schedulable processes) by the scheduler immediately, subjecting its pages to garbage collection (the garbage collector avoids pages being actively referenced by members of the balance set). Upon wakeup, the process in question would have to fault back any needed pages lost during the dismissed interval, experiencing delays and overhead and contributing to total paging traffic. The 1.33 scheduler now notes any process which dismisses for an interval less than a fixed system-global value, currently 100 ms, and holds that process in the balance set for a period not exceeding that value following all subsequent dismisses. This continues until a period of dismissal exceeds the threshold value. The purpose is, of course, to protect the working sets of such dismissed processes, on the evidence that the process is likely to awaken soon. This change has been observed (at Xerox PARC) to produce a notable reduction in paging traffic.

D. BCPL

1. PDP-10 Compiler

During the preceding quarter, a new set of PDP-10 code generators was written. These code generators were written to run with the same parser as the PDP-11 compiler.

This quarter, we continued work on these code generators, to make them fully functional, and to realize a 10 to 20% increase in compiled code efficiency. After studying the compiler, it was decided that the entire program needs careful re-organization in order to make it both more easily modified, and more easily transported.

A set of header files which describe all of the compiler data structures, lexemes, globals, and implementation constants has been created. The operator precedence scheme has been isolated into a single easily modified table, and the entire parser and tree-building (CAE) phases have been re-written and are undergoing testing. The tree-walking/code-generator phases have been examined, and the tree walking (TRN) phase has been re-coded to make use of the structure capabilities of BCPL.

2. BDDT

BDDT is a source level symbolic debugger for BCPL. In addition to fixing several bugs in BDDT, a new experimental version was

developed which enables users with all uppercase terminals to debug programs which use both upper and lower case letters. This version needs further work before it is ready to be released.

3. PDP-11 BCPL Cross-compiler

Further work on the PDP-11 BCPL cross-compiler was performed in this quarter to decrease the number of cases where the compiler cannot resolve conflicting register requirements resulting in abortion of the computation. The register use optimizer attempts to select the order of evaluation of operands in order to minimize the number of extraneous moves either in setting up the operands or in utilizing the result. The procedure is sub-optimal in that each operator is considered only once and the decision on evaluation order is done solely on the basis of the register usage in evaluating each operand. This results in a procedure whose cost (in compilation time) grows linearly with the number of operators involved rather than one which grows as 2**N.

IV. COTCO Activities

A. Background

As an initial part of the effort to consolidate communications on Oahu, NAVELEX is considering experimental use of ARPANET-supported interactive message services to supplement existing record communication systems. CINCPAC and NAVY personnel will evaluate the applicability of specific and general capabilities for possible use in the final COTCO system. The activities described below provide support for the planning and execution of such a test using existing ARPANET message technology. In particular, we will adapt and enhance current ARPANET message facilities for such a test, including appropriate modifications and additions to current message computer support systems as necessary for military test use. The system changes described will upgrade the existing message capabilities to a level suitable for the test and provide for enhanced reliability in order to assure that a meaningful test can be conducted. The improvements planned will also be useful to the new message software being developed at ISI through the provision of needed system support features.

The ARPA sponsored work in message systems is currently being studied by a Message Service Committee with representatives from the relevant organizations in the ARPA community. These organizations include BBN (Bolt Beranek and Newman) MIT-DMS (Massachusetts Institute of Technology Dynamic Modelling Systems), SRI-ARC

(Stanford Research Institute Augmentation Research Center), and USC-ISI (University of Southern California Information Sciences Institute). ARPA Has tasked this committee with:

1. Identifying possible requirements for current and future message services within DoD as well as the ARPA research community.

2. Identifying the computer science issues and research problems that must be solved to meet these requirements.

3. Evaluating current ARPA R&D efforts on message systems with respect to applicability to these services.

4. Generating short term, intermediate term, and long term plans for meeting the identified objectives.

The recommendations generated for ARPA by the Message Service Committee should prove helpful and illuminating in identifying requirements for a direct writer-to-reader service.

The existing ARPANET message service has been in continuous development by the ARPANET community for the last two years. We were responsible for developing these services on the TENEX operating system. This service is provided by four different program modules on the host computers of the ARPANET:

1. SNDMSG: The message-creator is a special purpose interactive editor program which helps a writer compose the text of his message and a header which specifies subject, action addressees, information addresses, etc. Once the writer is satisfied with the message, he gives a command for the message to be sent. At this point, the message-creator queues a copy of the message for background-mode transmission to each addressee.

2. MAILER: The message-sender is a system-provided background job which periodically surveys the outgoing message queues written by the message-creator. The message-sender is responsible for transmitting each queued message to its addressed destination, and removing each message from the queue after its receipt has been positively acknowledged from the receiving end. This transmission takes place in the ARPANET using a protocol called the File Transfer Protocol, (FTP)

which provides control and positive acknowledgement for message moving between hosts on the ARPANET.

3. FTPSRV: The message-receiver is another system-provided background job which is awakened as needed to receive, store, and acknowledge incoming messages from other ARPANET hosts. It uses the ARPANET standard FTP for control of message transmission. Once the message-receiver has received a message, it must interpret the address field to dispatch the message to the correct recipient. The message is delivered by appending it to the addressee's message-file.

4. MAILSYS: The message-reader is an interactive program module invoked by a message recipient to assist him in processing the received messages in his message-file. It provides facilities for surveying all received messages (display of selected header fields), reading the messages in any order or combination on his terminal, and producing hard copy of selected messages. It also permits the recipient to dispose of messages by deleting them or appending them to other files of his choice for later retrieval. The message reader will retrieve messages from these other files in the same way it

retrieves messages from his message-file, permitting selective retrieval based on author, subject, date or a number of other keys. This constitutes a primitive cataloged information storage/retrieval facility.

B. Project Activities

During calendar 1975, we will design and implement modifications and extensions to existing writer-to-reader message facilities on the ARPANET to support a test-system for NAVY/DoD message processing requirements. Existing software requires specific modifications to improve message service reliability, responsiveness, human engineering, generality, and accounting capabilities. These changes in the existing message service software will provide such things as appropriate military message header fields and formats. Changes in the supporting system software will correct known minor deficiencies and improve overall system reliability. The result of our efforts should have continuing value in that the system enhancement will be useful in general or other applications and many of the message service functions will form a basis for further extension by others such as ISI.

We started this activity in January 1975, and plan to have a usable initial message system available in first quarter 1975. This initial system, which will contain only minor changes from the

current systems to enhance reliability and other critical aspects of the system, will be evaluated by NAVELEX. A plan will be generated for utilizing such a system for the experimental test-system. We will then complete any relevant changes, respond to additional requirements discovered during the evaluation, and investigate the issues involved and needed developments for meeting long-term requirements for military message processing.

Message system programs will be implemented in BCPL (Basic Compatible Programming Language) a high-level language which is currently supported on PDP-10's, PDP-11's, IBM-360's, Honeywell 6080's, and an assortment of other popular machines. At some future date, it may be necessary or desirable to transport message system functions into other machines, and our BCPL structure will greatly aid this process.

Our efforts will adapt the existing message services to the military environment and formats and provide additional capabilities in the support systems needed for the test. The current ARPANET interactive message service is a promising basis which can be molded into a suitable test-system economically and rapidly. By adapting existing and proven software and by using already existing hardware and communication facilities, an initial test can be conducted at low cost and on a rapid time scale.

C. MAILSYS

A first experimental version of the new MAILSYS Message Processing System was put into operation the previous quarter. After a period of testing and refinement the new system was released for BBN TENEX users, in January 1975. This first version of MAILSYS provides message management services for incoming mail. The system contains commands for manipulating "in-box" and other message files. It is possible to survey the contents of a message file, to perform selective retrieval on stored messages, to delete unwanted messages, and to move messages from one file to another. The retrieval mechanisms make it possible to extract complete messages or any combination of message field that the user may wish. Selective retrievals can "filter" on the contents of various message fields, such as author, date, addressees, or subject.

Message creation in this first version of MAILSYS is handled by invoking the standard SNDMSG program from within MAILSYS. Another feature with very general powers is the ability to invoke a "inferior" Executive System. This permits, for example, file copying and other manipulations not directly provided for in the MAILSYS code.

Since the January release, substantial progress has been made on a new and more powerful version of MAILSYS. Work on the new version is nearing completion, and its release is expected in the near future.

As far as a user is concerned, the most important feature of the new system will be the ability to create and transmit messages from within MAILSYS without the need to invoke SNDMSG. The new message creation facility will be command driven rather than having the dialogue format of SNDMSG. It will permit the user to specify message fields in any order he wishes and to return to previously specified fields as often as desired for editing or replacement. The user need not transmit the message until he is satisfied with its entire contents.

A second change of direct significance to the user is the ability to treat a message retrieval "filter" as an object which can be used repeatedly and incrementally changed during a session with MAILSYS.

The new version of MAILSYS will also contain a series of incremental changes, improvements and extensions beyond the present version. They account for almost half of all changes currently requested by users and planned for eventual implementation in MAILSYS.

D. DOCUMENTATION

An initial user guide was released with the January version of MAILSYS. Work is under way on a revised version of this document which describes the February MAILSYS release.

The second type of user documentation consists of commands built into MAILS that provide explanatory assistance to the user. Initial versions of these commands were implemented in the January release of MAILSYS. These are now in process of update, and new versions will be available along with the next MAILSYS release.

E. MAILER and FTPSRV

The project calls for modification to these underlying service programs in order to support and enhance the operation of the evolving MAILSYS system. The majority of these tasks are intended to render the operation of MAILSYS more efficient and to provide for enhanced security. During the current reporting period we have completed modifications to MAILER which provide for the delivery of local message items through the FTPSRV subsystem. Modifications to FTPSRV provide for queuing of incoming messages when the destination inbox is busy.

F. TENEX Capabilities

In addition to MAILER and FTPSRV, efficient, safe, and secure operation of the new MAILSYS calls for certain changes in the

underlying TENEX operating system. A basic modification has been made in the TENEX accounting system that will permit message processing services to be charged to the user's logged in job number. Formerly, a certain portion of these were charged to system overhead jobs.