connections. More recently, we have been able to restore telephone dial-out service by upgrading our EtherTIP systems. Now users can connect to dial-out ports on the EtherTIP from their workstation directly (e.g., Macintosh) or from the SUN-4 to make external modem connections. Our X.25 service line from TELENET Inc. also is attached to an X.25/Ethernet gateway which provided users immediate access to the new SUN-4 system from TELENET (see Wide Area Networking).

Although we are still missing a number of features in the new system (e.g., adequate community and project directory and file management structures, usage accounting, a WHOIS data base, and a fully-operational archiving system), the physical aspects of transferring to the SUN-4 are nearly complete. During this transition, the 2060 has been maintained only on a "time & materials" basis as absolutely necessary to retain adequate access. Numerous parts of the system have failed in the intervening 9 months, including several memory banks, disk problems, and a backplane problem. The system is now quite unreliable but not worth repairing as its resale value is nearly $0. We hope to keep it running through this summer in order to complete the file system transfer to UNIX and to tie off access to old 2060-based software. At that time, the 2060 will be shut down permanently, ending an important phase in the history of SUMEX-AIM and the AIM community.

(3) The New SUN-Based SUMEX-AIM Resource

The SUN-4/280 central server for the SUMEX-AIM resource was acquired in 1988, is configured with 32 megabytes of memory and 1.8 gigabytes of disk storage, and runs under the UNIX operating system (see Figure 3). The primary function of this machine is to provide, in place of the 2060, wide- and local-area network access; electronic mail transmission/routing, reception, and user access; community bboards; file service; and print spooling for the AIM community. Such UNIX servers, running on modern high-performance hardware like the SUN SPARC chip or other RISC chips, are relatively inexpensive and fast and are easily obtainable by other groups in the AIM community.

We also operate a complementary SUN-3/180 machine, acquired in 1987, which is named KNIFE (KSL Network Interface and File server Environment) and which provides additional file storage and access facilities for the community (see Figure 4). Both SUN servers run the same version of the SUN Operating System (SunOS 4.0), providing the expected efficiency advantages from uniformity — the KNIFE server had been running the earlier release 3 operating system and was only recently updated to release 4.0.

A third VAX 11/750-based file server (SAFE) is being phased out in deference to the more cost-effective SUN systems and its disks will be reintegrated with SUMEX-AIM and KNIFE.

These servers provide distributed computing services, including NFS (Network File Access), to the various personal computers and workstations in the AIM community. SUMEX-AIM is the main EMail machine and provides individual mail services, as well as network distribution lists and bulletin boards. KNIFE, on the other hand, is more committed to network and distributed file service development and support. Having two such different yet similar environments has provided good flexibility in offering various functionalities, while easing the administration of the distributed systems. We anticipate exploiting this duality in the coming year, specifically in the area of file backup and archiving.

(3.1) File Access and Management

A stable, efficient mechanism for storing and organizing data is central to any computing environment, and is one of the most challenging issues in the move to distributed, workstation-based computing. It is necessary to provide standard services, such as file backup, archiving, a flexible and intuitive naming facility, and data interchange services (e.g., file transfer). Also, as the amount of data being manipulated grows, it becomes more and more important to have powerful tools for managing hierarchies of files.

UNIX has many of the needed facilities, e.g., backup, long names, hierarchical directory structure, some file property attributes, data conversion, and limited archival tools. However, while general issues of networking, remote memory paging services, and flexible file access have received considerable attention in both the academic and commercial development of file servers, there has been only slow development of other critical operational tools. For instance, the much-used file archiving system of the DEC 2060 (sometimes called off-line cataloged storage) has no analog service in "standard" UNIX systems. Perhaps this is the result of UNIX having its origin in the small computer world where the number of users and volume of data has traditionally been quite low. To ensure continued availability of archiving services in our transition from the 2060, we have worked on adapting a commercial system developed by UniTech to allow users to manage large file collections by moving files not needed on-line to and from off-line tape storage. This system also maintains a historical archive of files. See the section on the "SUMEX Perpetual Archive System" below.

We have had a well structured organization for managing disk usage and accounting on our 2060 system and we plan to duplicate some of these features on the SUN-4. A UNIX accounting system will be put in place to allow cost accounting and provide a quota enforcement capability on the distributed file servers. UNIX has always been weak in the management of large-scale file backup and with the advent of disks of near gigabyte size, it is clear that a better organized approach is essential. In support of the long-term goals of the distributed community, we have been reviewing more advanced data storage methods. Optical disk systems are attractive but have not progressed as quickly as many had predicted. However, it seems likely that this sort of equipment would be ideally suited to satisfying our requirements for the archiving of files. Helical-scan magnetic tape equipment might also serve this function well. However, in both cases the absence of industry standards introduces some level of risk when planning to use these types of data storage equipment. We plan to continue our investigations of these technologies this coming year and to make a decision.

CAP, the public domain Columbia AppleTalk Package, is installed on both servers. Within its facilities, the AppleTalk/UNIX File Service package (AUFS) provides icon-based file support on a UNIX server so that Macintosh workstations can connect and see representations of their UNIX files in the same format as with the file tools on the native Macintosh. File transfers are invoked by moving icons around the Macintosh screen just as if the UNIX server were an extension to the internal Mac desk top and disk. This system operates either through Kinetics FastPath AppleTalk/Ethernet gateways or through pure Ethernet connections.

(3.2) The SUMEX Perpetual Archive System

Overview

The SUMEX Perpetual Archive System is an integrated set of system software and operator procedures which allow individual SUMEX users to create and manage perpetual magnetic tape archives of their files and directories. Features include:

- A command which requests the system operator to move the contents of specific files or directories to the magnetic tape archive. Note that the files are first moved to an intermediate on-line repository with other files awaiting archiving, so unless a user specifically requests that the contents of a file be retained on-line, the disk space used by the archived files will be immediately removed from their on-line disk quota.
- A command to interrogate an on-line catalog of archived files.
- A command to request the retrieval of archived files. In the case where a file of the same name is entered into the archive more than once, the user may specify a date such that the version of the file entered into the archive just prior to that date will be retrieved.

Historical Perspective

The current implementation of the SUMEX Perpetual Archive System is actually the third such implementation at SUMEX. The first was based on the BBN TENEX operating system, and utilized the BSYS magnetic tape backup software package written by Smokey Wallace at the SRI Augmentation Research Center in 1974. This BSYS-based system was in place from 1975 until 1984, when SUMEX was upgraded from a dual-processor DEC KI-10 system to a DECSystem 2060 running the DEC TOPS-20 operating system — a direct descendant of BBN TENEX. By that time, DEC had integrated into TOPS-20 an archiving capability that used the regular magnetic tape backup program, called DUMPER, also originally from BBN. The DUMPER based system was place from 1984 to 1988, when SUMEX was once again upgraded, this time to a more distributed system based on a SUN-4/280 network and file server, running UNIX. The current system utilizes a third-party magnetic tape backup software package called UBACKUP from UniTech Software Inc., which in turn utilizes the standard UNIX "tar" magnetic tape backup program.

As the SUMEX computing facility has changed operating systems, first from TENEX to TOPS-20, and then from TOPS-20 to UNIX, facilities have always been provided to retrieve files from the archives of the previous system — hence the basis for calling this archive system *perpetual.*

Implementation:

Normally, when a UNIX account is created for a user, he is assigned a home directory on a file system (for example /a/user) and is assigned an allocation of disk space on that file system called the *quota.* At SUMEX, an additional directory, called the user's *archive directory,* is created on another file system, in this case /var/archive/user. Once the archive directory is created, any files moved into it will automatically be moved to the magnetic tape archives the next time the operator runs the archive utility. The frequency of moving files from the archive directories to tape is a function of the total amount of space allocated to all archive directories, but is guaranteed to be at least once a week (on Sundays), and not more than once a day.

Commands

The archive and retrieve commands have a similar format, and can be thought of as commands that move files and/or directories between a user's home directory, say /a/user, and archive directory, say /var/archive/user.

archive [-d] [-h] **pathname...** [-as **pathname2**]

retrieve [-h] **pathname...** [-as **pathname2**]

Key features of the *archive* and *retrieve* commands include:

- Multiple arguments and options are allowed, in any order, and wild cards will be expanded by the shell as usual.
- The *-d* switch specifies that files requested for archiving are to be deleted from the system afterward.
- The *-h* switch specifies that the specified files and/or directories are to be archived from or restored to the *top* level of the user's archive directory. For directories, the structure below the specified directory is preserved.
- The *-as* switch allows the user to specify a different name under which the file or directory is to be archived or restored.
- Either absolute or relative *pathnames* may be supplied, i.e. it doesn't matter where the user is connected at the time of issuing the archive

command (although the examples below all assume the user is connected to the home directory.

- In the case of directories, all subdirectories of that directory will be recursively archived.
- If a filename conflict occurs because the user requests a file by the same name be archived before the previous request has been processed, archive will warn the user and ask for confirmation.
- If a filename conflict occurs because the user requests a file by the same name be retrieved, and a file by that name already exits, retrieve will warn the user and ask for confirmation.
- When requesting the archive of files and/or directories outside the user's home directory, the standard UNIX protection scheme applies.
- Deletion of the archived files is left to the user.

Examples

In the following examples, it is assumed that the user is logged into his directory (username) and is connected there:

Archive request	**Archived as**
archive .login	/var/archive/username/.login
archive mail/mail.sent	/var/archive/username/mail/mail.sent
archive /a/username/mail/mail.sent	/var/archive/username/mail/mail.sent
archive -h mail/mail.sent	/var/archive/username/mail.sent

(3.3) Printing Services

Laser printers have long ago become essential components of the work environment of the SUMEX-AIM community with applications ranging from scientific publications to hardcopy graphics output for ONCOCIN chemotherapy protocol patient charts. We have done much systems work to integrate laser printers into the SUMEX network environment so they would be routinely accessible from hosts and workstations alike. This expertise has been widely shared with other user groups in the AIM community and beyond.

SUMEX operates 4 medium-speed (8-20 pages per minute) Imagen laser printers, 6 low-speed (~3 ppm) Apple laser printers, 1 low-speed (~3 ppm) Xerox laser printer, and 1 low-speed (~1 ppm) Apple color dot-matrix printer.

All of the Imagen printers incorporate an emulator for a line printer, a Tektronix plotter, and a typesetter (using the *Impress* language). Additionally, the two Imagen *3320* printers implement the *PostScript* typesetter language (also implemented by the Apple LaserWriters) required for printing Macintosh documents. The Xerox printer (an *8046*) interprets the *InterPress* typesetter language. The Apple *ImageWriter LQ* is a low-cost dot-matrix printer that the Macintosh's *Quickdraw* printer driver is capable of using for color printout.

IMAGEN host software is installed on both the SUMEX-AIM and KNIFE servers, providing print spooling services for client workstations to any of the several Imagen Printers in our computing environment. Since two of the Imagen printers (3320's) provide UltraScript support (a PostScript-like document description language), we have connected the CAP and IMAGEN software together such that Macintosh users can spool print jobs to the IMAGEN printers. These printers appear in the Macintosh *Chooser* the same as any local LaserWriter.

Since the NeXT computers print with the PostScript language, we didn't acquire any additional printers to support these workstations.

In total, the laser printers printed about half a million pages of output during the year. Most of the printout was simple text (primarily electronic mail listings) and documents formatted on the Macintosh — technical papers, drawings, program listings, and screen dumps. As was our intention, only a very small number of charts was printed on the ImageWriter LQ — complicated charts and graphs that benefitted from the use of color to distinguish otherwise indistinct graphic or data elements.

(4) Electronic Mail

Electronic mail continues as a primary means of communication for the widely spread SUMEX-AIM community. As reported last year, the advent of workstations has forced a significant rethinking of the mechanisms employed to manage such mail in order to ensure reliable access, to make user addressing understandable and manageable, and to facilitate keeping the mail software distributed to workstations as simple, stable, and maintainable as possible. We are following a strategy of having a shared mail server machine which handles mail transactions with mail clients running on individual user workstations. The mail server can be used from clients at arbitrary locations, allowing users to read mail across campus, town, or country.

The mail server acts as an interface among users, data storage, and other mailers. Users employ a Mail Access Protocol (MAP) to retrieve messages, access and change properties of messages, manage mailboxes, and send mail. This protocol should be simple enough to implement on relatively inexpensive machines so that mail can be easily read remotely. This is distinct from some previous approaches since the mail access protocol is used for *all* message manipulations, insulating the user client from all knowledge of how the mail is actually stored on the server. This means the the mail server can utilize whatever data storage and access methods are most efficient to organize the mail on a particular server system.

Thus, a user sitting in front of his personal workstation can read mail from any system on which he has an account and on which such a mail server is running. This has several advantages to the usual scheme of TELNETing to such a host and then reading the mail on the host, and receiving the text in a byte stream over the network in use. Given a MAP designed for efficiently accessing mail on such a network, the effective bandwidth for text transfer can be maximized in two ways. First, when mail is accessed, a compact representation of the users new mail can be sent for him to peruse and act upon. This information is essentially an envelope containing addressees and the subject of the message — the text of the message is sent only when the user decides to read it. Efficient clients can read ahead and cache envelopes while the user is perusing information and composing messages or when the client is notified of the arrival of new mail. This maximizes the users' *perceived* bandwidth since the protocol minimizes the delays between the receipt of useful information. Secondly, the MAP server knows it is dealing with mail and can maximize network bandwidth by maximizing the data in each packet it sends in reply to a MAP request. In our servers and clients we utilize these techniques whenever possible.

The first prototype Interim Mail Access Protocol (IMAP) was designed with this in mind. As noted in our previous report, the prototype IMAP server on the DEC-20 provided the foundation for the IMAP protocol version 2 (IMAP2). During this reporting period the IMAP2 protocol description was published as an ARPANET Internet working group Request for Comments[1], making the IMAP2 specifications available to the general ARPANET community. This has resulted in IMAP2 clients being developed elsewhere. In fact, such a client has been implemented in Japan by NTT to run on their ELIS lisp machine. The implementer regularly reads his mail in the U. S. from Japan using this client.

We completed several tasks noted in last year's report — the DEC-20 server and Xerox Lisp client were upgraded to IMAP2; we implemented an IMAP2 server for UNIX; and demonstrated a prototype client in Common Lisp for the Texas Instruments Explorer. In addition to finishing this work in progress, the main thrust of our effort this year was to write a mail client for the Macintosh. The Macintosh client will be in alpha test by early summer 1989. In addition, a UNIX workstation mail client, called MS, using the Columbia MM CCMD package, has been implemented with an MM-like command interface. The MS client was initially developed at SUMEX as a debugging tool since it uses a command line user interface rather than one driven by a mouse, menus and windows. MS also runs under MS/DOS on IBM PC's, and on NeXT machines. A NeXT client with a graphics interface is also well on its way, and is being developed at the University of Washington.

[1] Crispin, M. R. "Interactive Mail Access Protocol - Version 2." Internet Working Group Request for Comments No. 1064, Stanford University, July 1988.

(4.1) Macintosh client - MacMM

When we closely examined the design of a client, it became clear that a large portion of the code was machine-independent, and should be written in such a way that it could be ported to non-Macintosh systems. A client comprises a mail reading module, mail composition module, IMAP2 protocol module, SMTP (Simple Mail Transport Protocol) module, and a TCP-IP network communication module. The first two of these require extensive user interfaces. When we began this project, we decided to write it in C because of its availability on most machines. At that time Apple was alpha-testing an TCP-IP device driver, MacTCP, for the Macintosh, but a stream or socket interface to this code to facilitate simultaneous use by multiple applications was not available and had to be developed. It is important to note that MacTCP supports multiple active applications using TCP-IP simultaneously, and this is very important in our environment. Since the last three modules mentioned above could be written and debugged under UNIX, we started the project on two fronts.

A simple line editor reader/composer user-interface was written to run under UNIX. This was initially considered to be "throw-away" code, but as mentioned above, has evolved into three separate IMAP2 clients. The IMAP2 and SMTP modules were then written in C to be machine independent, as was a small interface module to the operating system-dependent TCP-IP package. Thus for each new client, one was required to write the machine dependent calls to the resident TCP-IP code, and the mail reader/composer modules. At the same time we undertook a project to write a high level stream interface to the Apple MacTCP TCP-IP driver that was in alpha release.

These two projects proceeded in parallel and were finished at about the same time. The UNIX-based client, MS, was then ported to the Macintosh in the simple line editor mode, interfaced to the TCP-IP code, and was used to debug the IMAP2 module. At this point, we began to write the graphics-oriented mail reader user-interface.

(4.2) Mail Reader User-Interface

We began designing the user interface (i.e. client) in mid-September 1988. Although we knew what commands we would have to implement (from our earlier work on the Xerox Lisp machine client), we had to maintain the "look and feel" of the Macintosh for each of these commands. Additionally we wanted to be able to run other applications concurrently with the mail system and to bounce back and forth between these applications. This implied that the data structures maintained by the client be kept to a minimum to avoid running out of memory. We also had to get a working understanding of the Macintosh system calls before we could start coding.

We began the actual coding in late October. The first task was to implement a status window to inform the user of various events (such as notifying the user that there are new messages) and also as a place to output debugging information. This is an informational window only. We also had to implement the basic menu system. Almost every application has at least three menus (the Apple, File, and Edit menus) and the mail system is no exception. The File menu contains global commands such as "Open Mailbox" and "Compose Message". The Edit menu allows the user to Copy, Cut (copy and delete), and Paste (insert) selected text. Although the Edit menu is primarily used in composing a message, users can also manipulate text in the read and status windows.

The next step was to implement a scrollable message header browser window which has a one-line descriptor (message number, date, sender, subject, and size of message in characters) of each message in the user's mailbox. This is the heart of the mail system in that the user must select messages in this window before anything can be done to these messages. For example, to read a message the user must first select the message to be read. To select a message the user just points the mouse to the message descriptor and clicks it. There are two menus associated with the browser window; one that operates on the selected messages and another that affects the entire mailbox. The message menu allows the user to Read, Flag, Delete, etc. the selected messages.

There is also a "Select" menu item that allows users to automatically select messages that have certain properties. The user specifies these properties by buttoning fields in a "dialog" box. The mailbox menu allows users to Expunge deleted messages, Check for new messages, and Zoom selected messages (replace the browser display with only those messages that the user has selected). As of this writing, the Zoom and Print (to hardcopy messages) commands have not been fully implemented.

Next we implemented Read windows and the associated Read menu. The Read menu allows the user to operate on the current message in the Read window. The user can simply continue on and read the next selected message or perform some action on the current message. These actions include Answer, Delete, Flag, etc.

The entire mail system is menu/window driven. By this we mean that the active window determines which menus are enabled and the contents of the active window determine which menu items (in the enabled menus) are enabled. For example, if the message browser window is active, then only the browser menus are enabled.

Additionally if there are no messages selected, then none of the items in the message menu are enabled (with the exception of the Select command) since these commands only operate on selected messages. This approach eliminates confusion since the user can see at a glance which commands are allowed and which are not. The initial release of Macintosh MM will only allow one open mailbox at a time and one read window. The code was designed to allow multiple mailboxes and read windows so this restriction will be removed soon.

(4.3) The Mail Composition User Interface

In late March 1989, we began the mail composition user interface. This interface is also mouse/menu driven and supports simultaneously opened composition windows. Each such window has two independent panes; the top pane is for the composition of the mail header information, and the bottom for the text of the message itself. Each pane has independent scrolling and fonts. The mail header can be checked for the correctness of RFC822 syntax at anytime, and any addresses in error are flagged by both an error message "alert" and cursor pointing to the portion of the text that contains the error. The mail header can be freely edited, much like one would do in Emacs, except that the field headers, e. g., "From: ", are protected in the sense that they cannot be deleted. Both panes support the usual Macintosh text editing features such as cut, copy and paste. At the time of this publication, one can compose and send messages, save and restore drafts of messages, and the composition user-interface is essentially completed. Also, one can tailor the mail composition environment to ones own needs.

For this customization to be accomplished, each Macintosh has a composition initialization file. Using this file one can set a personal name, default font, font size and font face, i. e., plain text, **boldface** or *italics*.. Also, a system administrator can set the default SMTP server name list, and the host from which the mail appears to be sent, and to whom the replies may be sent.

Several features need to be added and are currently in progress. Among these are mail forwarding, message reply-to and answering, and the queuing of unsent mail and background mailing of this same mail. It is expected that this list may grow by user demand during its alpha and beta testing phases.

(4.4) Texas Instruments Explorer Client

The IMAP2 client for the TI Explorer system using Common Lisp mentioned in last year's report is still being developed by not as intensively as the Macintosh client because of staff limitations, and the fact that the AIM "market" for the Explorer system is significantly smaller. Nonetheless, substantial progress has been made on this client by James Rice. The TI Explorer IMAP Client is in a partial state of completion. The basic functionality for sending, receiving and replying to messages is defined, as is support for BBoards. The major areas where more work is needed to make it usable are in the command handler; significant modification of the original prototype implementation is needed, it still has a number of bugs, and the interface with the Zmacs text editor works but will probably need modification. We estimate that an additional month of work will be required before the system is ready for alpha-test.

(4.5) DEC-20 IMAP2 Server

The current version of the DEC-20 server is a complete and robust implementation of IMAP2 and we do not anticipate any further work on it.

(4.6) UNIX IMAP2 Server

UNIX is becoming more and more widely adopted in the AIM community and in our distributed mail system we will rely heavily on UNIX engines as servers. As summarized in last year's report, we have written a UNIX IMAP2 Server (UIMS) for remote mail access. UIMS handles the full complement of IMAP2 commands, and has been vigorously tested for the past year.

This past year, we benchmarked UIMS against Columbia MMC on a SUN-3/180 running SUN OS 4.0.1 to compare speeds for text searches of reasonably large mail files. This was done by modifying the MMC code to calibrate searches. The UIMS already has calibration incorporated into its code. Note that this compares wall clock search time between UIMS and MMC, and in both cases file I/O is factored out. It is interesting to note that making these measurements for UIMS was tricky. This is because, unlike MMC which reads the entire mail file at initialization, UIMS is very careful to minimize its working set size in order to reduce paging overhead and UNIX scheduler penalties for programs with large working sets.

UIMS only keeps messages in memory if the search succeeds, or if the message number is within 40 of the number of messages in the mail file, or if the user has "shown interest" in the message prior to the search. So, two searches were done in UMIS. The first was for the string "a" which coerced all of the data into memory, and the second was for the string "ImNotInAnyMessage%%%". Thus, the second search is independent of file I/O.

The results of these measurements for various mail file sizes are summarized in the following table:

Mail File Size	MMC Search Time	UIMS Search Time
2.5 megabytes	44 secs	11 secs
1.2 megabytes	16 secs	2 secs
0.5 megabytes	4 secs	1 secs

Table comparing EMail text search times for MMC and UNIX IMAP

Note that 0.5 MB is approximately the typical mail file size on SUMEX-AIM, ignoring large users and the very large BBoard files. The data illustrate the efficiency that can be gained by accessing mail from a special purpose server which is optimized for access performance, independent of the user interface code which executes separately on the user's client workstation.

The principal development effort on the UIMS this past reporting period was the addition of code to allow flexible access to read-only bulletin boards. In addition, we fixed a number of bugs reported by client developers.

We believe that accessing mail in this client/server model will prove to be more efficient than the usual mode of using mail reading programs on the mail server itself, or using network file system protocols to remotely access mail (i.e., down-load the entire mail file to the workstation. In the former case, the operating system overhead for running a job under UNIX has much more impact on system resources, than does an UIMS server process. The latter has no shell process for executing commands, and was written carefully to minimize system resource usage. The example of the search comparison above clearly points out this difference. Secondly, network file system protocols, like SUN Microsystem's NFS, show a drastic decrease in performance as the distance between the server and client host increases. This is even true on a local area network, where a client and host are separated by one or more gateways. This is not evident with our mail client/server model because IMAP2 was designed to be transactional and minimize the impact of a server and client being on separate subnets of a local area network, or even being separated widely on the Internet.

(4.7) Transition Strategy and Plan

The transition strategy plan described in last year's report is well underway. The AIM community currently reads mail on the SUMEX-AIM SUN-4 using the Columbia University MMC system which closely emulates the earlier TOPS-20 MM interface. UIMS understands the MMC mail storage format and we can access the same mail on one of the several personal workstations mentioned above. By the end of this summer, we expect most users will be reading their mail using an IMAP client on Mac II or Explorer workstations during the day, and in the evening, they will use a TELNET connection to SUMEX-AIM from a terminal emulator to process mail with MMC.

It is again important to mention that Columbia University distributes MM-C with the statement "Permission is granted to any individual or institution to use, copy, or redistribute this software so long as it is not sold for profit, and provided this copyright notice is retained". This makes MM-C an ideal mail reading program for the SUMEX-AIM community since our move away from the DEC-20 toward a distributed environment, since this interim mail-related software can be made widely available to the national community.

(5) Lisp Systems

(5.1) Standards

In a heterogeneous computing environment, such as AI research inevitably involves, the issue of cross-system compatibility is a central one. Users of various machines want to be able to share software, as well as be able to use various machines with a minimum of overhead in learning the operating procedures and programming languages of new systems. Thus, it is crucial to specify and propagate powerful, flexible standards for various aspects of the computing environment so that it is possible to transfer both skills and information among machines.

In order to improve the inter-machine compatibility of our software, we have been encouraging all users to use the Common Lisp programming language[1], as well as pressing vendors to provide more complete and efficient implementations of this language. We have already served as beta test sites for Xerox, Texas Instruments, and Lucid Common Lisp implementations.

The Common Lisp language, however, is only a subset of the software needed for our research. Research projects need higher-level powerful facilities, such as an object-oriented programming system, sophisticated debugging and error handling tools, portable window systems, and better graphics interface development tools. Therefore we have been supporting and following the development of evolving standards such as the Common Lisp Object System (CLOS), Common Lisp X Window system (CLX), and proposals for higher level User Environments (e.g., CLUE) via membership in the electronic discussion groups and specific technical contributions.

(5.2) Lisp System Performance

One of the key issues in selecting the systems for our distributed computing environment was the performance of Common Lisp. In order to assist us in evaluating the performance of Lisp systems, we undertook an informal survey of Common Lisp environments using two KSL AI software packages, SOAR and BB1. The results have been compiled into a KSL Technical Report presented in Appendix B. In this survey we focused on execution speed for simulated runs of the test programs and for compilation of the test programs as measures of performance. However, we emphasize that execution speed benchmarks are only one aspect of the system performance evaluation, especially for Lisp systems. Other crucial issues like programming and usage environments, compatibility with other systems, ability to handle "large" problems, and cost must also be considered.

Both SOAR and BB1 were chosen because they are implemented in pure Common Lisp, making them extremely portable. SOAR is a heuristic search based general problem solving architecture developed by Paul Rosenbloom et al.[2] and BB1 is a blackboard problem solving architecture developed by Barbara Hayes-Roth[3]. Neither of these systems is an intensive user of

[1] Steele, G. L., Jr. *Common Lisp - The Language.* Digital Press, Burlington, MA, 1984.

[2] Laird, J. E., Newell, A., and Rosenbloom, P. S. "SOAR: An Architecture for General Intelligence." *AI Journal,* 33(1):1-64, 1987.

[3] Hayes-Roth, B. "A Blackboard Architecture for Control." *AI Journal,* 26:251-321, 1985.

numeric computation. They were initially developed in environments other than those tested and no attempt was made to optimize their performance for any of these tests.

The workstation systems to be tested were chosen based on their availability as well as projected applicability in AIM community environments. Since we were interested in "real world" results, we ran the tests on each machine in what seemed to be its standard operating mode. The machines tested include:

SUN 3/260 with Lucid Lisp1
SUN 3/60 with Lucid Lisp
Compaq 386 with Lucid Lisp
Compaq 386 portable with Lucid Lisp
SUN 4/260 with Lucid Lisp
SUN 4/280 with Lucid Lisp
DEC MicroVax II with VaxLisp
DEC MicroVax III with VaxLisp
SUN 3/75 with Franz Extended Common Lisp
Texas Instruments Explorer I
Texas Instruments Explorer II
Texas Instruments Explorer II Plus
SUN 4/280 with Franz Allegro Common Lisp
Hewlett Packard 9000/350
SUN 3/75 with Kyoto Common Lisp
SUN 3/75 with Lucid Lisp
Apple Macintosh II with Allegro Common Lisp
Symbolics MacIvory
Texas Instruments microExplorer
IBM RT/APC with Lucid Lisp
Symbolics 3645
Xerox 1186

A large variation was observed between the ranking of systems when running the SOAR test and when running the BB1 test. This leads us to conclude that while these experimental results, and ones like them, can be used to

---

Hayes-Roth, B., and Hewett, M. BB1: An Implementation of the Blackboard Control Architecture. In *Blackboard Systems*, Engelmore, and Morgan editor, Pages 297-313. Addison-Wesley, Palo Alto, CA, 1988.

class machines together roughly, it is impossible to use such a set of benchmarks to decide in advance how a given application will perform on a given system. There is no substitute for actually running the program on the systems in question. However, on the basis of these data, the systems tested may be ranked as follows:

- Very Fast (≤ 0.50 anr — averaged normalized run time): TI Explorer II Plus (Exp2+), TI Explorer II (Exp2), and SUN 4 with Lucid Lisp (4/280 and 4/260)
- Fast (> 0.50 anr, ≤ 1.00 anr): TI microExplorer (mX), Compaq 386 (386), SUN 4 with Franz Lisp (F-4/280), Compaq 386 portable (386T), SUN 3/260 (3/260), IMB RT/APC (RT), and SUN 3/60
- Medium (> 1.00 anr, ≤ 1.50 anr): Symbolics 3645 (Sym), SUN 3/75 with Lucid Lisp (L-3/75), HP 9000/350 (HP), TI Explorer I (Exp1), and DEC MicroVax III (DEC-III)
- Slow (> 1.50 anr, ≤ 2.50 anr): Symbolics MacIvory (Maci), SUN 3/75 with Kyoto Common Lisp (K-3/75), and SUN 3/75 with old Franz Extended Common Lisp (E-3/75)
- Very Slow (> 2.50 anr): Apple Macintosh II with Allegro Lisp (Mac2), DEC MicroVax II (DEC-II), and Xerox 1186 (XCL),

The data were normalized by dividing individual results by the average of the results for all the tested implementations.

As new Common Lisp implementations become available to us we plan to revise the report to include measurements of these systems. We are currently integrating measurements for Symbolics MacIvory running Genera 7.4 and Symbolics 3650 running Genera 7.2. We are working on getting measurements for Symbolics XL400, DEC MIPS, and Cray 2, as well as other mainframe systems.

(5.3) Lisp Programming Environments

Even though performance gaps between microprogrammed Lisp systems and stock workstation implementations are narrowing, there still remains a significant difference in the quality of the development environments. The power of the development tools and environment is what has been the primary strength of Lisp machines, allowing rapid design, implementation, and debugging of complex programs. We believe the key to good development tools is integration, both in terms of consistency of interface, and in the ability to move seamlessly from tool to tool, carrying along appropriate data and state information. These qualities must be manifest in any AIM research computing system.

Over the years, KSL and AIM community AI systems have been implemented predominantly in the InterLisp, MACLisp, ZetaLisp, and more recently, Common Lisp dialects. Beginning in the early 1980's, our work moved from mainframe Lisp environments to workstation environments for many reasons, principally involving powerful tools for system development and debugging and graphical interfaces. Commercial versions of these tools, that evolved over many years in the Xerox D-Machine, Symbolics 36xx, LMI, and TI Explorer systems, have become an indispensable part of our work environment. Newer Lisp systems for workstations not specifically developed for Lisp have lacked many important features of these environments.

Thus, in light of the runtime performance advances of stock workstations, we have attempted to summarize the key features of the Lisp machine environments that would be needed in stock machine implementations in order to make them attractive in a development setting (the first draft of this specification was included in last year's report). Unfortunately we have been unable to refine this draft specification over the past year due to lack of resources. We hope to better address the issue of high-quality development environments on a variety of platforms in the coming year.

(6) Workstation System Environments

(6.1) Macintosh II Workstations

Hardware and Software Environments

The installation of the Macintosh II workstations and ancillary networks, printers, and software went mostly according to plan, and most have been in productive use for over a year now.

The inexpensive Rodime winchester disks have proven quite reliable so far. The few units that have failed were replaced under warranty. A greater number of Moniterm displays exhibited a tendency to fail from overheating, with annoying but not problematic frequency.

We are using Microsoft's *Word* for word processing, *PowerPoint* for presentations, *Excel* for accounting, and Blue Sky's *TEXtures* and *LaTEX* for large specialty documents. After long deliberation, we have chosen Deneba's *Canvas* for technical illustration and Niles & Associates' *EndNote* for bibliography preparation. We have also been able to make good use of inexpensive file system repair utilities like Central Point's *MacTools*, Symantec's *SUM* and Alsoft's *Disk Express*. We bought a copy of *Mathematica* for the Mac II to tide users over until the faster free version becomes available for the NeXT machine.

We cooperated with some other laboratories on campus to purchase a site license for Coral's Allegro Common Lisp for the Macintosh. Although generally liked by the students who did small projects in this environment, this implementation was not employed in any big projects this year. To test Allegro Common Lisp's capability as a systems language we implemented a small utility that coalesces multiple lines of a TOPS-20 text file into unified paragraphs suitable for Microsoft Word's consumption. The overhead for using such a utility was relatively high — about the same as HyperCard's — but not unacceptable.

After we purchased the site licence (as did Carnegie Mellon and MIT), Apple Computer's Advanced Technology Group purchased the rights to Allegro Common Lisp and plans to continue to improve it and use it in their own research and distribute the improved version through APDA. We believe that this development will work to our advantage in the long term, making Apple Common Lisp a good, inexpensive vehicle for small AI applications.

We bought two copies of Connectix's *Virtual* software and Motorola PMMU hardware to experiment with virtual memory on the Macintosh. The product is mostly successful, but currently has problems with MacTCP and Allegro Common Lisp. Although basically successful, we have decided to defer purchase of more virtual memory hardware pending support by Apple and because of the cost.

MacTCP Stream Interface Package

Beginning last August, we alpha and beta-tested MacTCP — Apple's implementation of the lower levels of the Department of Defense's TCP/IP networking protocol suite. (The product manager for MacTCP is a former SUMEX employee now at Apple).

Before MacTCP, no TCP/IP-based applications for the Macintosh allowed simultaneous access to the network interface by other applications. Thus, one was limited to running a single network application at a time.

In writing Macintosh programs, accessing a driver is very complex and requires subtle calls to Mac OS primitives. For example, it requires 150 lines of code to open the MacTCP driver, create a low level I/O stream, and then open this I/O stream for reading and writing — and much of this code just defines "the bits" for a parameter block which is passed to the driver via a system queue. To eliminate these difficulties we wrote a high level stream interface to MacTCP — *tcpio. tcpio* permits one to easily access a remote host with five simple function calls: *tcpopen*, *tcplisten*, *tcpread*, *tcpwrite*, and *tcpclose*.

Tcpopen opens an active connection to a remote host, while *tcplisten* waits passively for connections from remote hosts. This latter allows one to run daemons in the background, e.g., a *finger* daemon which reports the status of the current user (if any) of a Macintosh. Each returns a TCP stream handle which is used in subsequent calls to *tcpio* functions.

Tcpread and *tcpwrite* read and write data on the open tcpstream, and *tcpclose* closes the connection. If one wishes to use asynchronous I/O—i.e., read and write without blocking—then three additional functions are provided: *tcplistencheck*, *tcpreadcheck*, and *tcpwritecheck*.

One calls *tcplistencheck* to see if a remote request for a passive connection has been made; *tcpreadcheck* to see if a read has been completed (and if so, how many bytes were read); and *tcpwritecheck* to check if a write has completed.

All of these functions return complete error messages in the case of an OS or network failure of some sort.

The marriage of tcpio and MacTCP provides excellent bandwidth on a local area network and will allow a single application to open up to sixty-four simultaneous TCP connections. One can read/write large blocks of data in excess of 400 kilobits/sec between a Macintosh and SUN-3 with both hosts on the Ethernet, and in excess of 150 kilobits/sec between the same two hosts when the Macintosh is on a LocalTalk network. Given that the bandwidth of LocalTalk is 230.4 kilobits/sec and that we use a Kinetics bridge between the LocalTalk and Ethernet, this performance is remarkable.

Domain Name Service

This spring MacTCP was in beta release and domain name service was added to the driver package. For reasons similar to those mentioned above, we wrote a high-level interface to this package. It allows one to do host name and address lookups, and uses a disk resident host file for familiar and frequently looked up names and addresses.

This package returns fully qualified domain names with each query, so, for example, if one looked up the name *SUMEX-AIM*, not only would one receive its network address—36.44.0.6—but also "SUMEX-AIM.Stanford.EDU," in the reply. The converse is true for address probes. If a host has more than one address associated with its name, then all addresses are returned in the reply to a name lookup.

MacTCP - tcpio Applications

The above two packages are prerequisites for the writing of network applications on the Macintosh and are at the heart of MacMM, the mail reader/composer we are developing for this system.

An early first use of this package was to allow the Guardian project to distribute its analysis and results between Macintosh's and an Explorer. The Guardian BB1 program ran on the latter system, and sent data to Macintosh's for further analysis and graphics display.

We are also interested in doing remote database queries from the Macintosh to our SUN file servers. We have provided our tcpio package to Sybase so that they can provide TCP/IP connectivity from their Macintosh client to the Sybase database server that runs on SUN's.

Our tcpio package is in the public domain and will ultimately be distributed with MacTCP by Apple along with other public domain software that has been developed internally by Apple.

(6.2) Texas Instruments Explorers

The Texas Instruments Explorers have enjoyed an increasing popularity as more projects have developed a need for the combination of execution speed, full Common Lisp, and sophisticated development facilities offered by the

Explorer. The KSL use of Explorers has expanded to include 6 Explorer II's, 28 Explorer Is, and 16 microExplorers, for a total of 50 Explorer family systems. Our efforts have been directed at improving the environment of the Explorer by developing software, organizing user interest activities, and advising Texas Instruments.

Previous experience has shown that the greatest source of advancement for a particular computing environment is the user community. They are the most in touch with the deficiencies of the system, and thus uniquely positioned to address them, as well as to utilize the strengths of the system. The product developers of the system are frequently too involved in the lower levels of detail to produce general, effective solutions to problems, as well as being hampered by limited manpower resources. However, a significant amount of time and effort is required to organize this effort. This task has traditionally fallen to a user-run organization, such as DECUS or USENIX.

We have spearheaded an effort to organize an international users' group for the Explorer. The slightly misleading name for the group is NExUS, standing for National Explorer Users' Group. In the past year the NExUS steering committee has drafted a mission statement for the group, held a general meeting in conjunction with the AAAI conference in St. Paul, moved towards a generally accessible library of user-contributed software, negotiated a relationship with TI-MIX, the general Texas Instruments users group, and maintained an electronic mailing list of several hundred people which has distributed over 400 messages among Explorer users world-wide.

We have maintained and extended the software tools produced in the KSL and made available to the national community. The tools now include:

36XX-EXPLORER

BACKUP-TO-FILE-SYSTEM

BATCH-PROCESSOR

DEVELOPMENT-TOOL-CONSISTENCY-ENHANCEMENTS

DVI-PREVIEWER

EXPLORER-36XX

GENERAL-INSPECTOR

GRAPHER

GRAPHICAL-VALUE-MONITORS

IMAGEN-PRINTER-VIA-TCP

INSPECTOR-ENHANCEMENTS

KSL-PATCHES

MAP-OVER-FILES

MAP-OVER-FILES

PATHNAME-EXTENSIONS

SEARCH-AND-REPLACE
SEARCH-AND-REPLACE
SINGLE-WINDOW-VT100
SOFT-KEYS
SOURCE-CODE-DEBUGGER
SPELLING-CHECKER
STRUCTURE-ENHANCEMENTS
UTILITIES
WEST-COAST-WINDOWS
WINDOW-ACCELERATORS
WINDOW-DEBUGGER-ENHANCEMENTS
WINDOW-ICONS
WINDOW-SYSTEM-ADDITIONS
ZMACS-ENHANCEMENTS

Many of the tools have been enhanced or newly written this year.

GENERAL-INSPECTOR	The most significant of many improvements to this tool is the introduction of a "perspectives" mechanism, allowing the user to view a single data type (e.g. a list) as a representation for many different types of abstract data structures. For instance a list might be a mapping between tags and values implemented by a list of tag, value pairs (a Plist) or implemented by a list of sublists, the first element of each sublist being a tag and the second being the value (an Alist). Both example implementations have value in certain contexts, but often the user viewing such a data structure only wants to see what tags there are and what their values are. The perspective mechanism provides this. Also for example, a symbol might represent a named structure, a flavor, a CLOS class, a function, a type, a package, etc., and a perspective tailored to each of these is provided, as well as ways for users to conveniently add their own perspectives. This tool has also been extended to encompass the Common Lisp Object System and Portable Common Loops (using the perspectives mechanism).

WINDOW-DEBUGGER-ENHANCEMENTS	This tool has been extended to use the General Inspector.
ZMACS-ENHANCEMENTS	New features in this tool include: commands for formatting and pretty printing text and programs; commands for manipulating software systems defined with DEFSYSTEM; and more commands for dealing with Tag tables.

Of course, all of these will be provided to the user's library, and many of them have already been given to other sites, including IntelliCorp, Berkeley, ISI, University of Maryland, and Ohio State.

Third party software is less utilized, but we stay abreast of the latest releases of the expert system shell KEE. We have installed a DVI previewing system from MIT allowing TeX and LaTeX output to be viewed on the Explorer screen. We have available several other imported tools such as a Common Lisp LOOP package and we are experimenting with a Domain Name System Resolver.

In addition to producing and maintaining these software tools, we attempt to provide extensive testing and evaluation of Explorer hardware and software products in a sophisticated university research environment in order that these products work more effectively when they are distributed to the national community. This testing is critical to the development of the computing environment since the combination of concentrated in-house expertise and close links to the product developers allows a turn-around on problem fixes unavailable in the broader scope.

This year we have participated in testing TI's high-end product, the Explorer II-Plus, System Software Releases 4.0, 4.1, 4.1.1, 5.0, and 5.0.1, and the Common Lisp Object System. Many of these releases were focused on the relatively new microExplorer add-in co-processor for the Apple Macintosh II. Many of the suggestions we have made to TI in the course of using this machine have been integrated into the system. It is our hope that this work will result in high-performance, low-cost sophisticated Lisp availability, allowing greater dissemination of AI software to the national community.

TI's planned release date for their implementation of the emerging Common Lisp Object System (CLOS) standard has been moved forward by several months due in part to our urgings. Implementation of this standard will allow developers to write sophisticated, portable programs in an objected oriented framework.

In addition to specific testing and evaluation, we are constantly finding, tracking, fixing, and reporting software bugs. This year we submitted twenty-eight new bug reports on Explorer system software, almost all of which had fixes included. All of these fixes have been made available to the national community in a patch file.

As well as working on these specific problems, we have had many meetings with Texas Instruments representatives wherein we have attempted to present the needs of the national community for short- and long-term AI workstation products, covering issues including the desirability of specialized hardware, address space, programming environment versus execution speed, and the ability to utilize the AI workstation's power for routine tasks.

Of course, there is also a large number of day-to-day activities needed to keep the computing environment pleasant, including resource management (e.g., disk space allocation, printer management), assistance with file backup and magnetic tape usage, and introducing new users to the system. We have produced documents targeted at complete novice users, users of InterLisp-D machines, and users of Symbolics machines in order to facilitate user education. These documents have been used as examples at various places in the national community.

For the coming year we plan to continue development and maintenance of the software tools, perhaps adding facilities such as a DARPA Internet Domain Resolver, better IP access control, CLX and CLUE packages, better documentation, better software management facilities, a Zmacs novice mode, and an Explorer version of the TALK program, as well as aiding the growth of the users' group. We already have under development an Explorer IMAP mail client, UNIX LPD-based print spooling, and automatic file backup.

(6.3) SUN Workstations

Due to the high performance relative to purchase cost, SUN workstations have drawn strong interest as Lisp engines. For the past two years we have had three SUN 3/75 workstations in experimental use in the KSL. Because these were purchased for LISP work, we have added a 24 megabyte memory board (from Parity Systems Inc) to each of these. Also added were 70 megabyte SCSI disks. The systems have been set up to swap the large virtual memory to the local SCSI disk, rather than over the Ethernet to the server.

One of the systems is used for system development, a second has been used in the "Very Large Reusable Knowledge Base" project and a speech recognition box has been connected to another of the clients. An interface between the latter system and a Xerox InterLisp workstation running ONCOCIN was developed developed to study the use of speech input for medical information. An evaluation of SUN workstations was made in terms of their suitability as a platform for a general physician's workstation which would support data management, analysis and display, and consultative software. For now, the SUN/UNIX environment was judged not to be competitive, in terms of cost or user interface technology, with other workstations environments (e.g., the Mac II) for this purpose.

The vendor plans for LISP support on SUN workstations has been in a state of flux this past year. Despite the decision to stay with Lucid as a supplier of the underlying Lisp implementation, SUN has not made great progress towards providing a programming environment competitive with older Lisp machine systems. However, SUN is continuing to work in this area and has the potential to close the gap, and thus deserves watching.

(6.4) NeXT Workstations

This year we obtained four NeXT workstations for evaluation and development, funded half by the SUMEX-AIM grant and half by other sponsored research sources in the KSL. We started following the NeXT closely in the middle of last year with several of the KSL planners attending pre-announcement, non-disclosure sessions about the machine. Several laboratory members attended the official, day long NeXT announcement and three of the SUMEX staff attended the following "NeXT Day" technical session. In early January, one of the staff attended the four-day NeXT Developer's Course and we received our first two machines. One of these machines was allocated for system integration purposes to a staff member and the other was made available for evaluation by laboratory members. A few weeks later our second pair of machines arrived, one of which was allocated to an Medical Computer Science application (OPAL) and the other installed in our Welch Road facility for use in the core AI research work of the Heuristic Programming Project. All machines were configured with 8 MB of memory and 330 MB SCSI Winchester disk drives, along with the standard NeXT 256 MB optical disk drive.

To accommodate the NeXT machines, we installed new thin Ethernet in parallel to our existing thick Ethernet in two of our locations. We then (with the assistance of technical notes from the AIR/SPUDS organization at Stanford) configured the networking software to be compatible with our environment. Rather than introduce four new file systems onto our network that would require backup and maintenance, we used the internal disks for the system software and swapping, but user directories were mounted via NFS (Network File System) from our existing SUN-3 file server, KNIFE. For authentication and host name resolution, we made the NeXT workstations "Yellow Pages" clients of our SUN server. This eliminated the need for separate account creation procedures for the new machines and made it possible for any user to use any of the NeXT's interchangeably. This did introduce some unique problems of designing user UNIX initialization files (.login and .cshrc) in such a way that they were both compatible with NeXT and SUN logins. Once the networking was in place we then got remote printing working using both our generic PostScript printers and our Apple LaserWriters, by means of our SUN-4 server as a spooling host.

To help users get started on the machine, we set up a local mailing list, "KSL-NeXT", to distribute information, which now has nearly fifty readers, several of whom are outside the KSL. We obtained access to the nation-wide NeXT-News mailing list and a campus-wide mailing list, NeXT-Info. We made available copies of every technical document that related to the machine, including "Objective-C" and "PostScript". We received permission for several of our programmers to audit a class conducted on campus by NeXT about programming the machine. To further assist in learning to program the machines, we collected numerous non-trivial example programs with sources from public archives on the Internet and from the NeXT Developer's Course and made these available on-line.

Shortly after gaining some experience with the machines, we set up a meeting with Cindy Larson (sales representative) and David Bessemer (support engineer) from NeXT to discuss our status, progress, problems, and needs in using the NeXT machines. NeXT requested this meeting to learn about what we were doing with the machines and what was needed from them in the way of additional support and changes in future system releases. This meeting was quite successful and we have strongly influenced them regarding the need to fully integrate Common Lisp into their environment, particularly in the *Interface Builder* and *Application Kit*, and to make extensive use of the Common Lisp object standard in doing so. Also, we have sent in several dozen bug reports regarding the 0.8 release of their software. We are anticipating receiving the 0.9 system software release sometime this month.

We developed a pair of programs to control the display brightness and sound volume on the machines. These were both short-comings of the initial software release that we knew would be fixed in a later release but they were sufficiently annoying as to warrant some programming effort. These were well-received by the national NeXT community which was suffering with the same problems. Additionally, we developed a pair of conversion filters to facilitate moving sound files between the NeXT and the Macintosh. This allows us to use the built-in sound facilities of the NeXT to generate sound files for use in Macintosh applications, like *HyperCard* stacks. We wrote a bitmap conversion program to allow us to move bitmaps from our Xerox environment to the NeXT. Additionally, we are importing software from other universities, including both NeXT-specific applications as well as general Unix software such as *CSound* from MIT. We brought up the terminal-based, MM-like MS client for Unix, made available by Mark Crispin, which uses the IMAP protocol and software developed here. We anticipate Crispin making available a graphic mouse-based version built using the NeXT's Interface Builder, around the same time as we release our initial Macintosh version. We also installed a pre-release of the CMU portable speech library which included two speech recognition demonstration programs, *Sphinx* and *VSC* (voice spreadsheet). These are discussed in the Speech Project part of the report.

The OPAL project has made significant progress with the machine as the NeXT object-based interface construction kit fits naturally with that project's need to generate sophisticated interfaces programmatically. The machine has potential for use in our remote graphics and distributed computation work due to its Display PostScript server, which is accessible via Ethernet using TCP, and its Mach ports for remote procedure execution. This potential should start to become a reality with the next release of system software.

The machines have had an unexpected positive impact on the Speech project due to both their ability to run the CMU speech software as well as provide faster cycles than the SUN 3 currently being used to process speech using the SSI software library.

We had two early hardware failures, one of which was minor and both of which were repaired under warranty; all four machines are no longer under warranty. We have no plans to obtain more NeXT machines but their numbers continue to increase campus-wide and in the AIM community.

(6.5) Xerox D-Machines

This year saw the introduction of a new release of the Xerox Lisp environment, a new hardware platform, a new release of server software, and a new company. The use of Xerox workstations in the KSL continued to decrease though it has not yet disappeared completely. Most of the SUMEX-AIM systems software projects on these machines consisted of aids to transfer to other platforms and small prototype systems. As part of our removal of support for the equipment, we turned over to Ohio State University responsibility for the national INFO-1100 Xerox Lisp interest Internet mailing list which we have been operating since the machines first appeared seven years ago. We also discontinued the Internet software repository for Lisp programs on SUMEX-AIM,in conjunction with the SUN 4 transition, and this was picked up jointly by Xerox and Ohio State.

As mentioned very briefly in the previous annual report, we were involved early in the evaluation and testing of a new hardware platform, known as Maiko, which consisted of a Xerox Lisp byte code emulator, implemented on a SUN workstation in 'C' with some optimization of the machine code. The byte code emulator provided binary compatibility with the Xerox hardware thus making porting of software trivial. We used the ONCOCIN system to evaluate the new platform. Using the full range of SUN 3 and SUN 4 processors we saw performance ranging from half of the Xerox 1186 to several times faster than the Xerox 1132. Although technically quite impressive, in general we did not consider this as a means to prolong our use of the Xerox Lisp environment, due to the cost of the SUN hardware necessary to get the performance we would require. Maiko was seriously evaluated as a possible short term fix for some performance and packaging problems in both the clinic ONCOCIN system and the speech project, though later rejected. Xerox also saw this as a short term solution to the problem of moving onto more conventional hardware.

To facilitate the new platform, Xerox provided a new software release, Medley, for which we were involved in two separate beta-tests. We were one of two sites involved in the initial beta-test and this consisted of the usual testing and fixing of existing software and reporting of problems. For the most part, the new release was a better version of their previous combined InterLisp and Common Lisp release, Lyric, with just a few feature additions. This release provided increased support and faster throughput for TCP/IP-