

NATIONAL INSTITUTES OF HEALTH
 DIVISION OF RESEARCH RESOURCES
 BIOTECHNOLOGY RESOURCES PROGRAM

SECTION I - RESOURCE IDENTIFICATION

Report Period: From August 1, 1975 to July 31, 1976

Grant Number:

RR-00785-03

Report Prepared:

May, 1976

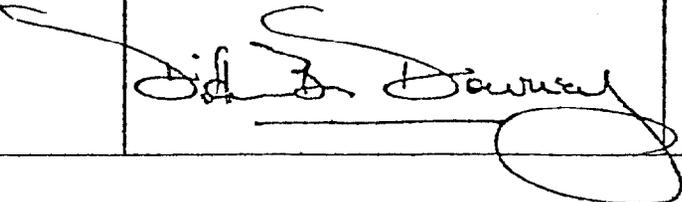
Name of Resource: Stanford University Medical Experimental Computer (SUMEX)	Resource Address: Stanford University Stanford, California 94305	Resource Telephone Number:
Principal Investigator: Joshua Lederberg, Ph.D.	Title: Chairman and Professor Department of Genetics	Academic Department: School of Medicine Department of Genetics
Grantee Institution: Stanford University	Type of Institution: Private University	Investigator's Telephone No.: (415) 497-5801

Name of Institution's Biotechnology Resource Advisory Committee:

SUMEX-AIM Executive Committee

Membership of Biotechnology Resource Advisory Committee:

<u>Name</u>	<u>Title</u>	<u>Department</u>	<u>Institution</u>
Saul Amarel, Ph.D.	Chairman and Professor	Computer Science	Rutgers University
Donald Lindberg, M.D.	Professor Director	Pathology Information Science Group	University of Missouri School of Medicine

Principal Investigator: Joshua Lederberg, Ph.D. Chairman and Professor	Signature: 	Date: May 25, 1976
Stanford University Official: D'Ann B. Downey Sponsored Projects Officer	Signature: 	Date: May 27, 1976

II RESOURCE OPERATIONS

II.A PROGRESS

II.A.1 RESOURCE SUMMARY AND GOALS

The SUMEX (Stanford University Medical EXperimental computer) project is a national computer resource funded by the Biotechnology Resources Program of the National Institutes of Health (NIH-BRP). It encompasses a dual mission: 1) the promotion of applications of artificial intelligence (AI) computer science research to biological and medical problems and 2) the demonstration of computer resource sharing within a national community of health research projects. The SUMEX resource resides administratively within the Genetics Department of the Stanford University Medical School and serves as a nucleus for a growing community of projects, both at Stanford and nationally. SUMEX provides computing facilities specifically tuned to the needs of AI research and communication tools to facilitate inter- and intra-group contacts and the demonstration of research products to medical users. The project also develops tools for and encourages community relationships among collaborating projects and medical researchers.

User projects are separately funded and autonomous in their management. They are selected for access to SUMEX on the basis of their scientific and medical merits as well as their commitment to the community goals of SUMEX (see Section II.A.3 on page 44). Currently active projects span a broad range of application areas such as clinical diagnostic consultation, molecular biochemistry, belief systems modeling, mental function modeling, and instrument data interpretation (see Section IV on page 68).

Artificial Intelligence is a branch of computer science which attempts to discern the underlying principles involved in the acquisition and utilization of knowledge in reasoning, deduction, and problem-solving activities(1). Each authorized project in the SUMEX community is concerned in some way with the application of these principles to medical problems. The tangible objective of this approach is the development of computer programs which, using formal and informal knowledge bases together with mechanized hypothesis formation and problem solving procedures, will be more general and effective consultative tools for the clinician and medical scientist. The exhaustive search potential of computerized hypothesis formation and knowledge base utilization,

(1) Two recent reviews give some perspective on the current state of AI: see Nilsson, N.J., "ARTIFICIAL INTELLIGENCE", Information Processing 74, North-Holland Pub. Co. and a summary by Buchanan, B. G. and Feigenbaum, E. A., attached as Appendix A, page 166. An additional overview of research areas in AI is provided by the outline for an "Artificial Intelligence Handbook" being prepared under Professor Feigenbaum by computer science students at Stanford (see Appendix B on page 180).

constrained where appropriate by heuristic rules or interactions with the user, has already begun to produce promising results in areas such as chemical structure elucidation, diagnostic consultation, and mental function modeling. Needless to say, much is yet to be learned in the process of fashioning a coherent scientific discipline out of the assemblage of personal intuitions, mathematical procedures, and emerging theoretical structure of the "analysis of analysis" and of problem solving. State-of-the-art programs are far more narrowly specialized and inflexible than the corresponding aspects of human intelligence they emulate; however, in special domains they may be of comparable or greater power, e.g., in the solution of formal problems in organic chemistry or in the integral calculus.

Our community building role is based upon the current state of computer communications technology. While far from perfected, these new capabilities offer highly desirable latitude for collaborative linkages, both within a given research project and among them. Several of the active projects on SUMEX are based upon the collaboration of computer and medical scientists at geographically separate institutions; separate both from each other and from the computer resource. Another major goal of the network experiment is to enable diverse projects to interact more directly and to facilitate selective demonstrations of available programs to physicians and medical students. Even in their current developing state, such communication facilities allow access to the rather specialized SUMEX computing environment and programs from a great many areas of the United States (even to a limited extent from Europe) for potential new research projects and for research product dissemination and demonstration.

This past year has seen much activity and growth in the SUMEX-AIM resource and community. Two new formal projects (one maturing from an earlier pilot effort) have been authorized to join the AIM community as well as several new pilot efforts. These new projects together with the increasing load from the earlier projects have raised the loading of SUMEX-AIM beyond productive limits, particularly during prime time. To accommodate this load, we received authority from the Executive Committee to augment the processing capacity of the system - implementation of this addition is in progress. Efforts have continued to build inter- and intra-group interactions through system communication facilities, workshops, a local "mini-conference" on AI techniques to pull together the Stanford community of projects, and a seminar project initiated by Professor E. A. Feigenbaum of Stanford to assemble from the community a handbook of AI concepts, methods, and state-of-the-art. There have also been continuing, substantial efforts by the community to introduce non-affiliated research people to a number of the programs which are far enough along in their development. The system staff has worked hard to maximize the computing resources and to enhance the repertoire of software available to the community and has investigated a variety of alternatives related to the import and export of operational programs. And finally, the management committees which help direct the allocation and development of the resource have been active in recruiting and evaluating new projects, planning future AIM workshops, and guiding the dissemination of resource objectives and opportunities to other medical institutions.

II.A.2 TECHNICAL PROGRESS

II.A.2.a FACILITY HARDWARE

Memory Swapping Drum System:

The hardware system has stabilized nicely over the past year, especially after the correction of a design flaw in the DEC-supplied drum controller(2). We had been having an abnormally high number of drum errors, mostly recoverable. The number exceeded the manufacturers' specifications and could not be explained by memory overruns or other contention problems. After much detective work (and vendor interactions), we discovered that a delay in the DEC drum controller between "drive select" and "sector ready" signals was too short to allow the read and write heads to settle down. After putting in the appropriate delay circuitry (in September 1975), the system has run to date without a single error (recoverable or permanent) or failure in the swapping system!

Computational Capacity:

A major event over the past year relating to system hardware was the decision to upgrade the central processor capacity. An updated diagram of the hardware configuration is shown in Figure 1 on page 9. As discussed in the last report, the high loading of SUMEX-AIM during prime time has restricted work. From a subjective viewpoint, the system became unworkably sluggish above a load average(3) of 4 or 5. We have made a number of administrative efforts to push as much work as possible into non-prime time. These have included excellent cooperation from user projects in encouraging programmers to work during night hours, doing operational functions (such as file system dumps) during the evening, and providing an effective batch system for running jobs in background mode and during non-prime hours. These steps have helped make better use of the non-prime hours but have not substantially relieved the midday congestion; the decreases achieved were offset by new development work and increased community use of operational AI programs (ONET, DENDRAL, MYCIN, and PARRY in particular). The principal period during which medical collaborators can, in practice, work with the programs remains the prime hours and our goal is to provide a computational capacity which allows reasonably interactive access to the programs at these times.

(2) We follow a long tradition in calling our fast, fixed-head disk a "drum" to distinguish it from the file system disks

(3) The "load average" signifies the number of jobs waiting in queue to be processed at a given instant: it measures the number of people awaiting service at that moment, so that responsiveness will be (approximately) inversely related to the load average. Two, three, or even four times as many users may be connected to the system at such times; but users typically take time out to ponder what the computer has reported, or the jobs may be preoccupied with input or output rather than the CPU.

We made a series of measurements to identify system bottlenecks(4) and observed a number of simultaneously limiting resources. A plot of the elapsed time required to accumulate 1 CPU minute is shown in Figure 2 on page 10 as a function of load average. A plot of the corresponding system I/O wait and core management overhead time is shown in Figure 3 on page 11. Data are plotted for a variety of jobs ranging from a one page CPU-bound job to a large, page-fault intensive jobs and include several FORTRAN and INTERLISP jobs. The data were not collected on a dry machine, they were run at night when the user load was low but not zero. Thus, some dispersion exists in the data. The key features of the data (and other internal system parameters) are that for small jobs the elapsed time-to-completion increases linearly with load average. That is at load average N, the CPU is split into roughly 1/N equal parts. For larger jobs, the low load average behavior is a linear increase in time-to-completion with load average but with an offset in elapsed time. This reflects the substantial waiting time (for a given job) to swap pages in and out. The I/O wait curve shows much dispersion at low load averages depending on the character of the system load. If there are not many jobs to be run, and one becomes unrunnable because of waiting for swapping or disk I/O, the wait time will be very high (see the upper curve of Figure 3). On the other hand, for the small, CPU-bound limit, the I/O wait loss is negligible (lower curve in Figure 3).

At load averages above 3 or 4, a non-linearity sets in for large jobs because of memory limitations and the increased swapping load (relative wait time, interrupt service, etc.) on the system. In the same limit the I/O wait approaches 15-20%. Of the 512 "core" memory pages currently on the system, almost 380 are available to users. With the current working set (5) limitation parameter (maximum 150 pages), 2-3 large jobs and up to 5 or so mixed jobs may be resident at once. If more than this number of jobs are runnable, some must be totally out of core and receiving no service. Because it is costly to move whole working sets in and out of memory (5-10 milliseconds per page), the system attempts to minimize "thrashing" by approximating a batch mode of scheduling, giving more run time before forcibly removing one program from memory to run another. This degradation is spread around evenly but causes added swap and core paging time in order to run a job and hence increases the per job time to complete.

However, based on user comments at loads of 3-5, the runtime increase because of load average (even without any additional overhead) becomes excessive as well - jobs that ran in several minutes at low load average begin to take tens of minutes at higher loads making interactions much more cumbersome. Another factor is that by the time there are more

(4) These bottlenecks refer to program execution. File storage has been another limiting factor for the system and is discussed later (see page 7)

(5) The working set is a group of pages which is a subset of the total active memory used by a program and which the system "guesses" (based on previous running history) will be addressed during the next running time quantum. In this way the system attempts to keep only those pages needed at any point for a program to execute during its time slice.

than 6 large jobs on the system, we begin to run out of drum swapping space. The current capacity is 3300 pages and, allowing for the monitor, can hold 5-6 full 256K work address spaces. Typically with a load average of 4-5, there are many more jobs on the system (25 - 30) with a range of memory requirements. Thus under such loads, the drum can accommodate even fewer large LISP jobs. Of course when the drum fills, swapping overflows to the much slower moving head disks and contends as well with regular disk I/O traffic. This substantially increases the system I/O wait time. As noted on page 12, we have implemented a page migration facility which assures that drum space is used only by active pages; but under heavy load we may still exceed the capacity.

From these data it is clear that with the typical mix of jobs on SUMEX-AIM including many large LISP jobs, above a load average of 5 or so the system runs out of memory, CPU, and swapping space at about the same time. Because of budgetary constraints we are not able to augment all three resources at once however. FOR A GIVEN LOAD, the effect of adding memory and swapping storage would be to linearize the response curve (Figure 2) through a reduction in system overhead at load averages above 4. For load averages in the range of 5-6, this could recover up to 15-20% in elapsed time/CPU minute. The augmentation of processor capacity to first order reduces the overall slope of the curves in Figure 2 and thus benefits users at all levels of loading. If the load is truly interactive (jobs complete or require terminal input after a few minutes), any speed-up in running time will tend to reduce the load average as well since the jobs will leave the run queue sooner. For many long, CPU-bound jobs this effect doesn't exist and the load average would stay the same - the overall run times for the jobs would be reduced however. In the ideal case, doubling processor capacity would improve elapsed time/CPU minute by 50%. This cannot be realized in fact since having a faster processor with the same memory means that the swapping rate will increase and hence total overhead will go up.

Because of the greater advantage for interactive jobs and because of budgetary considerations, the strategy approved by the AIM Executive Committee was to augment the CPU capacity as the first step, taking note of the certain need to augment memory and swapping storage soon thereafter. In addition to the technical arguments, the rationale for the decision also takes account of the fact that DEC CPU prices have been rising recently whereas memory prices are presumably still falling.

We examined both an upgrade from the KI-10 to a KL-10 and the installation of a dual processor KI-10. From a technical viewpoint, our preference was to upgrade to the KL-10, particularly in light of DEC's indication that the machine would be configured (microcode) within the year to efficiently run TENEX. For essentially economic reasons, however, the KL-10 option was not feasible. DEC marketing has taken a firm stand about selling KL-10's as "systems" which means that we would have had to upgrade not only the CPU but disks, tapes, and data line scanner as well. The net cost of upgrade would have been in excess of \$500,000 - well over our budget. In view of this and the feasibility of a dual processor system based on our studies, we decided to add a second KI-10. The implementation of this plan is now underway and proceeding very well - we are about ready to bring up the new system for user testing and hope to have it ready for use during the next AIM workshop this June.

It must be pointed out that at the time the decision for a dual processor was made (fall of 1975), we realized that the trend within the ARPA TENEX community could be toward a DEC-supported TENEX system although DEC had not made clear its plans for TENEX support. At this time that indeed seems to be the long term prognosis. DEC has announced the KL-20 (2040) running TOPS-20 which is a direct descendant of TENEX. The KL-20 is not a fast enough machine to have benefitted us in upgrade (it is slower than the KI-10) and delivery will only start in volume next fall. The rest of the KL-20 series of machine has not been announced although two bigger machines (currently denoted 1080T - to be called 20??) may be delivered to ARPA contractors late in 1976. A substantial amount of work remains to bring the DEC TENEX system up to the state of the current KA and KI TENEX systems which will likely take another year at least. Thus whereas in the long term the dual KI TENEX system will diverge increasingly from the DEC mainstream (by current projections), the pace with which it is coming into operational status and the minimal disruption to on-going user work, support the correctness of the decision relative to the pragmatic needs of the existing community.

We had expected delivery of the second KI-10 in late March but because of scheduling problems within DEC, we did not receive it until April 15. Because of additional delays in getting the needed memory and I/O bus cables to connect the new machine into the existing system, it could not be checked out to begin software development until the last week in April. Once installed, the machine has worked with only a few minor problems which were quickly corrected. Software development has gone equally well and we were ready to bring the full dual processor system up to begin user testing as of May 16.

Disk File Storage:

As mentioned earlier, the system has been operating at file system capacity as well over the past year. We have implemented policies which regularly clean out the file system (expunge deleted and temporary files as well as archive old files) to keep user projects within allocated limits. Nevertheless, many of the projects face severe constraints in available on-line storage needed for large LISP program development and community interactions. Because the system is fully allocated, there is little we can do to alleviate the problem within the present hardware. We are implementing operational improvements as possible to facilitate file archiving and restoring. We have also investigated the Datacomputer facility managed by the Computer Corporation of America over the ARPANET as well as other sources of on-line storage (at less loaded network sites) which could be available for a fee. The space available at the Datacomputer has been disappointing up to now. Some projects are trying out storing files at other sites but because of ARPANET access constraints, this is likely not a useful long term solution for the whole SUMEX-AIM community.

Since augmenting the file system is presently beyond our budget (higher priority is being given to improving computing capacity through CPU and memory enhancements), we have encouraged user projects which are

particularly cramped for space to assist in budgeting disk augmentations for themselves and the community. The DENDRAL and Chemical Synthesis projects each have proposals to cover additional RP-03 drives. We have two slots left on the existing controller and incrementally this is the lowest cost way to augment file space. Another highly attractive approach is using a System Concepts SA-10 "IBM" channel with double-density "3330" drives. However the initial cost of the channel, new controller, and one or two drives would be about \$100,000.

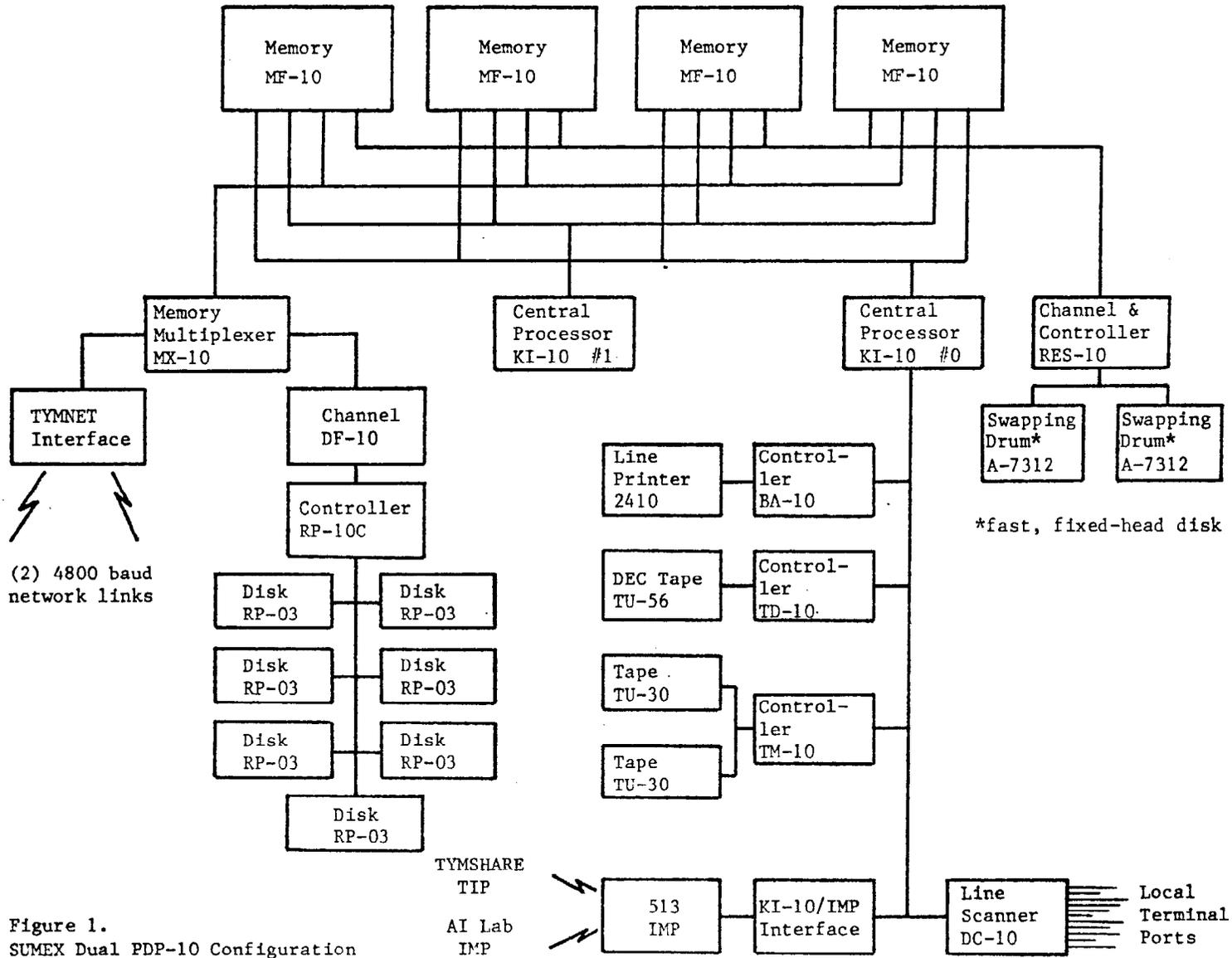


Figure 1.
SUMEX Dual PDP-10 Configuration

Figure 2.
SUMEX RESPONSIVENESS UNDER LOAD

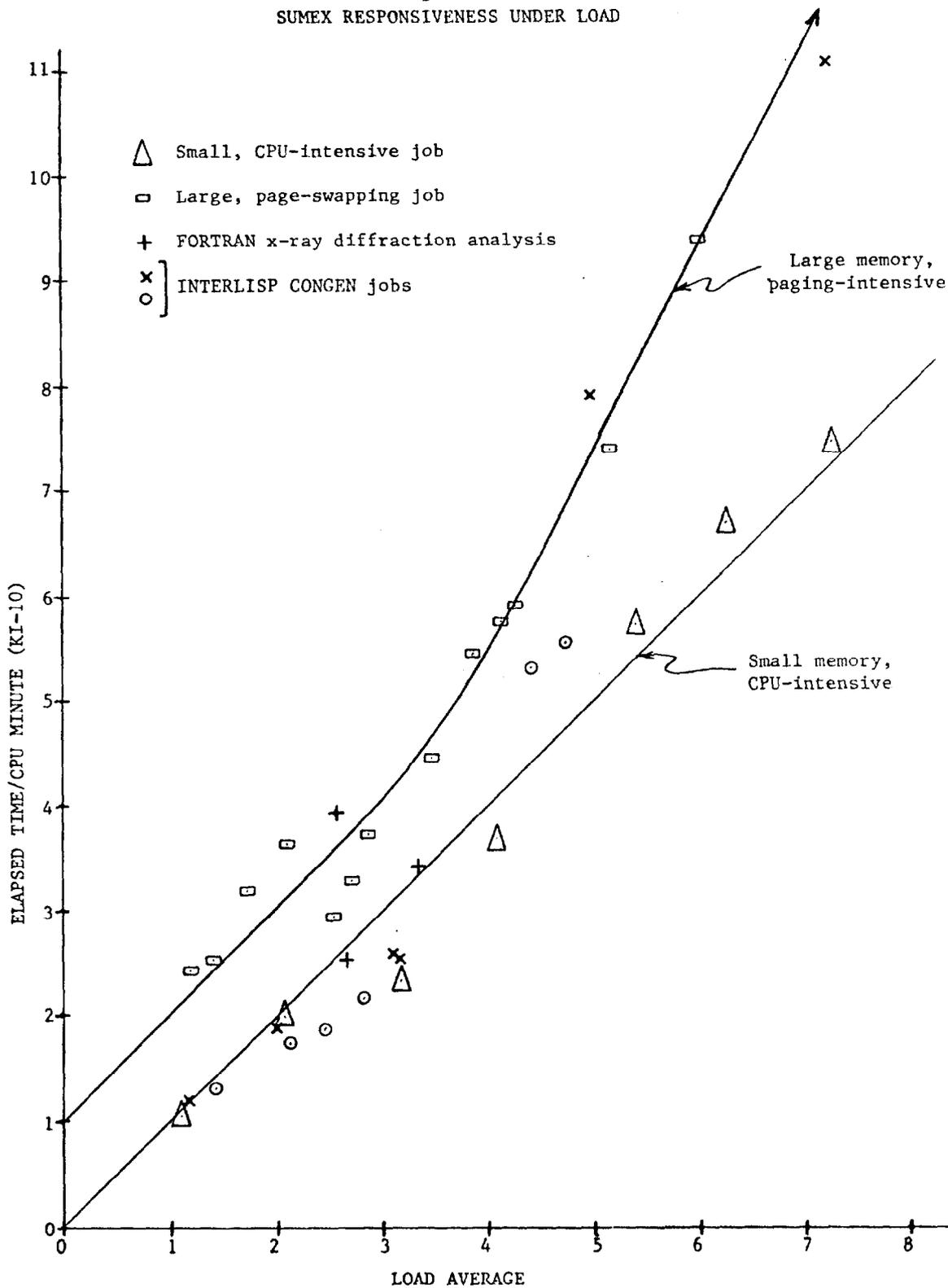
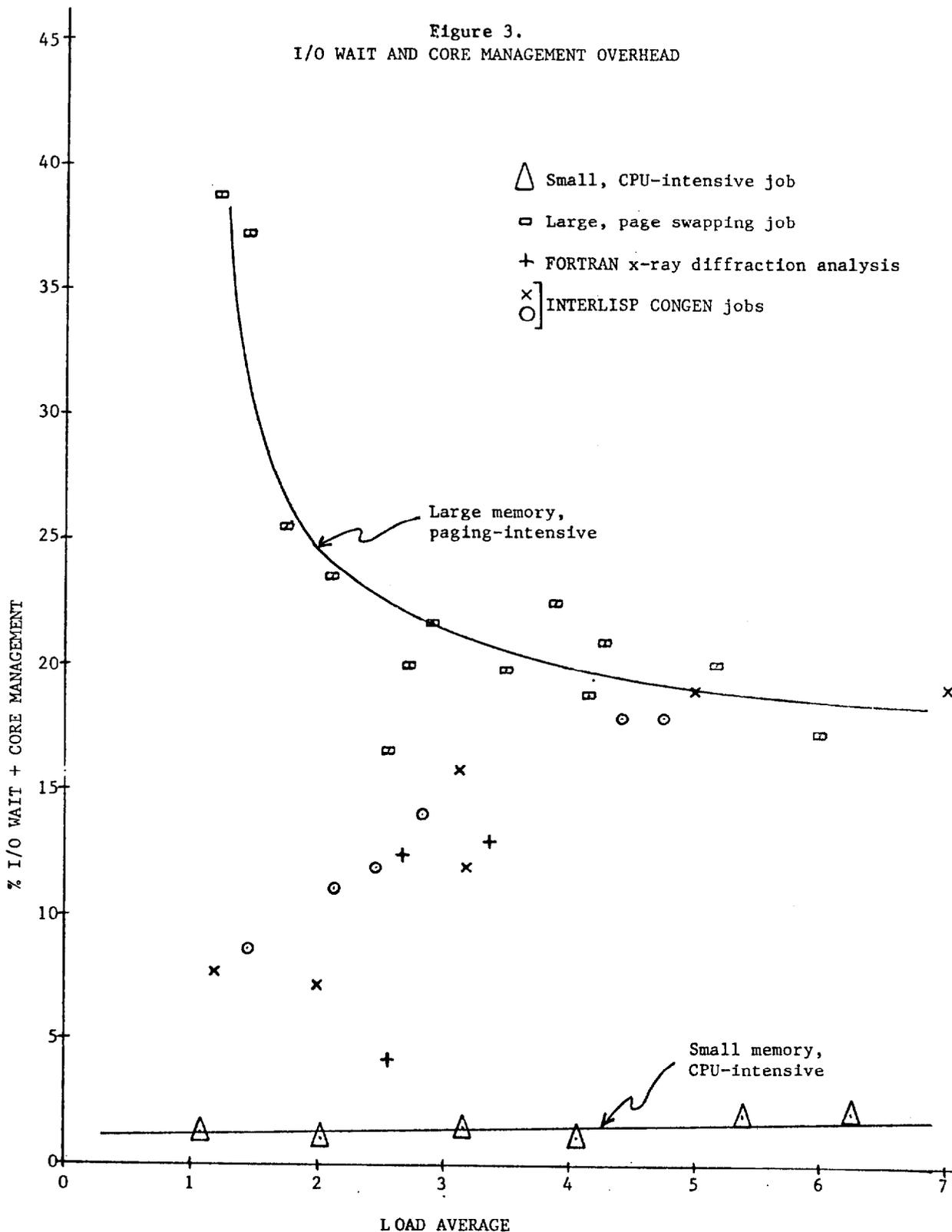


Figure 3.
I/O WAIT AND CORE MANAGEMENT OVERHEAD



II.A.2.b TENEX SYSTEM SOFTWARE

Our goals to improve resource allocation control capabilities, improve guest facilities, and maintain system compatibility with other ARPANET TENEX sites notwithstanding, we have continued to run TENEX release 1.31 this past year. The decision to upgrade to a more current release was deferred pending the decision on the processor augmentation. Had we been able to work out the acquisition of a KL-10, the monitor development effort would have had to take a different course to intercept DEC's monitor development efforts directly. Since that was not possible and a second KI-10 was approved in December 1975, we have begun the conversion to TENEX 1.33/1.34 in concert with the dual processor development effort (see below for a description of the trade-offs between the older 1.33 release and the very recent 1.34).

Swapping Storage Management:

Earlier in the year, while waiting for the processor decision, we finished implementing a drum page migration system which ensures that the drum is used only by active (recently accessed) pages. This optimizes the use of swapping space and reduces the substantial overhead when swapping overflows to the disk which is 5-10 times slower and contends with other file I/O. The garbage collector operates cyclically (currently every 10 minutes) and if a page allocated on drum has not been accessed during the previous interval and if alternative space is available on disk, it reassigns the page to disk. The cycle time of 10 minutes was chosen to give a reasonable time for a program to get around to using a page before declaring it dormant and at the same time not to penalize swapping of newly created pages by forcing them to reside on disk too long. This cycle time seems to be acceptable as we observe migrations of several hundred pages per cycle on the average. This has eliminated the situation where users first on the system leave dormant jobs around on drum and users who login later have their job pages allocated on disk because no drum space is free. Of course, during really peak loads the drum space may still become saturated with active jobs. We find that aggregate I/O wait times approach 40-50% with significant amounts of disk swapping whereas using drum, the I/O wait falls to under 20%.

Dual Processor Development:

Since the dual processor decision, the plan of attack has been: a) develop the dual processor system for KI-TENEX 1.31 which has been a thoroughly debugged system in our environment, b) in parallel, to transfer local TENEX changes (drum handler, TYMNET service, special JSYS's, etc.) to TENEX 1.33/1.34 and debug it as a single processor system, and c) after the dual processor system has stabilized and TENEX 1.33/1.34 is running well, complete the upgrade of our TENEX 1.33/1.34 to the dual processor configuration. The sequencing of these changes is designed to get the added capacity on line as soon as possible (particularly for the second workshop at Rutgers the first week in June) and to minimize the impact on users.

The dual processor software system is in the process of final implementation and checkout by R. Schulz and B. Hasselblad. We expect to bring the system up for user testing by mid May, conduct a detailed performance evaluation after the workshop in June, and complete documentation in July. Thus the following is only a preliminary report on the overall design philosophy. We will detail the system implementation and performance in the next report. From the start the design emphasized treating the two machines symmetrically and has maintained the ability to run the system either as a single or dual processor. The processors operate independently using common monitor code and system status data, each scheduling jobs independently, executing system calls, etc. The coordination between the machines is through status information in the data base and a set of interlocks which each machine can test and avoid simultaneous interference in sensitive areas. There have been many difficult issues in constructing the system of monitor interlocks and in debugging sections of monitor code for dual processor operation. This work has been greatly aided by the highly reentrant nature of the initial TENEX monitor design. The dual processor design has remained stable from its initial conception and implementation is proceeding on schedule. The detailed design began in early February with final design and implementation being done during late March and early April. After hardware installation during late April, debugging began. We began user access to the system on a test basis on May 16.

TENEX Monitor Upgrade:

Approximately 6 man-months of effort are being expended to upgrade the Tenex operating system at SUMEX from version 1.31 to 1.33/1.34. Version 1.32 was skipped because it was primarily a maintenance release which contained no new features or capabilities that we desired or required, although certain bug fixes and efficiency improvements were incorporated as deemed beneficial. We have been running version 1.31 for some 3 years. The major portion of work involves the incorporation of local SUMEX features into the new version including the dual processor changes, with the ensuing checkout and documentation phase.

Version 1.33 has been out in the field since January, 1975 and is a well proven and reliable system. It includes numerous bug fixes and improvements in efficiency along with a number of new features, the most important of which is the inception of the pie-slice scheduler. Version 1.34 is the most current version but has not been running as long. It has further updates to the pie-slice scheduler, bug fixes, and a reorganization of the source code. A final decision about whether to go to the proven 1.33 or immediately to 1.34 will be made before the end of May. We expect to have the new system up and running by late July.

The pie-slice scheduler provides system administrators with a mechanism for dividing user communities into groups ("pie-slices") and establishing minimum service levels for each group. These minimum service levels are guarantees which are met by the system regardless of activity in other groups. It is possible, of course, to observe a level of service in excess of the guarantee. This may happen either as a result of a group

being explicitly assigned the unclaimed share of an unrepresented group (the so-called "windfall") or simply as a result of small system load; no cycles are ever deliberately discarded. This represents a radical departure from the basically "laissez faire" 1.31 philosophy. In particular, at SUMEX where we have three somewhat separate user communities, a) local Stanford users, b) national AIM users, and c) SUMEX staff, we will be able to explicitly assign relative priorities of 40-40-20 respectively for the three groups and have the Tenex scheduler dynamically enforce them.

Completion of the 1.33/1.34 conversion effort has a few other implications. First, the dual processor software was developed as a parallel effort, so those changes will have to be incorporated into 1.33/1.34 as well. Second there is a new version of the EXEC (1.53) that goes along with 1.33 that has a number of new features and takes advantage of the new monitor JSYS calls provided by 1.33. Third there will be a reasonable documentation effort required, although most of the new features and commands are already documented in machine readable form, and only need to be put together in a suitable package. Fourth, there is the consideration of moving right on to Tenex 1.34. This version ties the core management functions of the operating system in more closely to the pie-slice scheduler, and would no doubt be beneficial in our environment.

In any case, the target is to have 1.33 (or 1.34) up and running long enough before September so that we can be sure we have a stable system, and any problems that arise can be ironed out during the summer.

TENEX Executive Upgrade:

Another area of software development is in the Executive program which is the basic user interface to manipulate files, directories, and devices; control job and terminal parameter settings; observe job and system status; and execute public and private programs. As mentioned above under "monitor", significant upgrade work here was delayed pending the decision on system augmentation since the TENEX upgrade affects the EXEC as well. As with all system work, we face a dilemma which is particularly strongly felt in this area; should we run a "standard" system or should we adapt things to user community needs and thereby tend toward a "home-brew" system? This is a difficult issue in that in many respects the SUMEX community is special - it includes a broad spectrum of users from professional computer scientists and programmers to biomedical research scientists and clinicians. The latter group, of course, want a minimum impedance to using the performance programs they are interested in while the former group wants a rich assortment of system facilities and as much flexibility as possible. Since most systems are designed for the programmer community, we have adopted the viewpoint that controlled augmentations of the system must be made to accommodate the medical user. Much of this work is still in process and will be for some time. The key point of this effort is to introduce knowledge about the individual user into the system (such as his usual defaults in using system functions, his level of expertise coupled to on-line assistance, his domain of interest to alert him to new information and perhaps personalized system commands

or macros convenient to his needs) so that he perceives a system tailored to his style and conventions in using the computer.

At this stage such information is stored and used from special files on a per program basis as in the MSG message reading program and TV editor. The EXEC has built in a number of parameter and subcommand setting commands which can be initialized by the LOGIN.CMD file. We will continue to devote effort in this area in up-coming work particularly to try to design a more uniform system pathway to such user-specific data.

Other EXEC command changes have been introduced to improve user interactions with the system (some developed locally and others designed by BB&N). These include commands for setting version retention specifications for files, purging (delete and expunge) individual files, improved system status displays, mail checking, TENEX error number interpretation, running programs explicitly as ephemerals (separate, transient address space) or non-ephemerals, and a group of SET and SHOW commands for various status conditions. One particular feature assists managing the large number of user written and supported programs that are available to the community. To keep these programs separate from the system-supported programs, another directory was created. Since the EXEC routinely searched only <SUBSYS>, <CONNECTED>, and <LOGIN> directories to find a program, it would miss all the user supported software. To solve this problem and give each user added control, we implemented a search path facility that is user settable. This allows each user to specify (with the SET PATH command) up to six directories (and their order) for the EXEC to search to find a program. The path is initialized by the EXEC to <SUBSYS>, <USESYS>, <CONNECTED>, and <LOGIN>.

Other changes will be forthcoming with the upgrade to TENEX 1.33/1.34 and the corresponding EXEC 1.53. These will include better ^C handling to solve type-ahead problems, a facility to have the EXEC periodically run a program for you (mail check, calendar check, etc.), system status displays accommodating the pie-slice scheduler, better human engineering in various areas, and a number of bug fixes.

System users can find up-to-date information on EXEC features through the EXEC manual and various on-line files:

```
<BULLETINS>NEW-EXEC.INFO
<DOC>TENEX-EXEC-MANUAL-UPDATE.INFO
<BULLETINS>LOGIN-CMD.BBD
<DOC>TENEX-133-EXEC.CHANGES
```

II.A.2.c NETWORK COMMUNICATION FACILITIES

A most crucial aspect of the SUMEX system is effective communication with remote users. In addition to the economic arguments for terminal access, networking offers other advantages for shared computing such as uniform user access to multiple machines and special purpose resources,

convenient file transfers for software sharing and multiple machine use, more effective backup, co-processing between remote machines, and improved inter-user communications. We have based our remote communication services on two networks - TYMNET and ARPANET. These were the only networks existing at the start of the project which allowed foreign host access. Since then, other commercial network systems (notably TELENET) have come into existence and are growing in coverage and services. The two networks to which we are currently connected complement each other; the TYMNET providing primarily terminal service with very broad geographical coverage and unrestricted user access, and the ARPANET having more limited access but providing a broader range of communication services. Together, these networks give a good view of the current strengths and weaknesses of this approach.

From the user's viewpoint, the reality of using a remote computer as if it were next door depends singularly on achieving the perception that a network connection is like a local telephone call to the computer. Current network terminal facilities do not quite accomplish the illusion of a local call. Data loss is not a problem in network communications - in fact with the more extensive error checking schemes, data integrity is much higher than for a long distance phone link. On the other hand, networking has as its underlying principle that through shared community use of telephone lines, widespread geographical coverage is possible at substantially reduced cost.

TYMNET:

Networks such as TYMNET are a complex interconnection of nodes and lines spanning the country (see Figure 4 on page 19). The primary cause of delay in passing a message through the network is the time to transfer a message from node to node and the scheduling of this traffic over multiplexed lines. This latter effect only becomes important in heavily loaded situations; the former is always present. Clearly from the user viewpoint, the best situation is to have as few nodes as possible between him and the host - this means many interconnecting lines through the network and correspondingly higher costs for the network manager. TENEX in some ways emphasizes this conflict more than other time-sharing systems because of the highly interactive nature of terminal handling (e.g., command and file name recognition and non-printing program commands as in text editors or INTERLISP). In such instances, individual characters must be seen by the host machine to determine the proper echo response in contrast to other systems where only "line at a time" commands are allowed. We have connected SUMEX to the TYMNET in two places as shown in Figure 4 so as to allow more direct access from different parts of the country. Nevertheless, based on delay time statistics collected over the past year from our TYMSTAT program, the response times are not very acceptable. The aggregate data are statistically summarized in Appendix D on page 195 and plots of the response time over the past year for particular nodes where we have extensive data are shown in Figure 5 on page 20. When delay times exceed 200-300 milliseconds, the character printing lag problems become noticeable with a full duplex, 30 char/sec terminal. These times have been particularly bad in New York with peak

delays approaching 3 seconds one way! Other nodes show uniformly high readings as well. These data reflect the subjective comments of many of our user groups as expressed in their individual reports (see Section IV on page 68). Problems have been particularly acute for Dr. Safir's group in New York and Dr. Amarel at Rutgers (see page 131).

We have had numerous meetings with TYMNET personnel to try to ease these problems and have instituted reroutings of the lines connecting SUMEX-AIM to the network. Also local lines to more strategic terminal nodes have been considered for users in areas poorly served by the existing line layout. These remedies have not had substantial effects. In general the TYMNET design goals are not to provide much better service. To quote from the April 1, 1976 TYMNET User's Group Newsletter:

"Current delay experience is from 0.25 to 3 seconds for a character to make a round trip through the network, with an average of 1.2 seconds. By early next year, "Clusters" will be installed in high density areas and will be interconnected by 9600 BPS lines. The result will be that round trip response/delay time will be less than 1 second for 80% of the cases and less than 2 seconds for 98% of the cases. This also is the design objective for TYMNET II."

We will continue to pursue improvements in TYMNET response but user terminal interactions such as used in TENEX programs are not realized in the time-sharing systems offered by most other TYMNET users and hence are not supported well by TYMNET. With these delays, it is not clear how well the proposed 1200 baud service they are going to inaugurate will work.

ARPANET:

The ARPANET, while designed for more general information transfer than purely terminal handling, has similar bottleneck problems in its topology (see the current geographical and logical maps of the ARPANET in Figure 6 and Figure 7 on page 21). These are reduced by the use of relatively higher speed interconnection lines (50 K baud instead of 2400 - 9600 baud lines as in TYMNET) but response delays through many nodes become objectionable eventually as well.

We are enforcing a policy to restrict the use of the ARPANET to users who have affiliations with ARPA-supported contractors and system/software interchange with cooperating TENEX sites. The administration of the network passed from the ARPA Information Processing Techniques Office to the Defense Communications Agency as of July 1975. At that time policies were announced restricting access to DoD-affiliated users. We have protected the facilities for calling from SUMEX out to other sites on the ARPANET to authorized users. This also protects the SUMEX-AIM machine from acting as an expensive terminal handler for other

machines - this function is better fulfilled by dedicated terminal handling machines (TIPS). In general, we have developed excellent working relationships with other sites on the ARPANET for system backup and software interchange - such day-to-day working interactions with remote facilities would not be possible without the integrated file transfer, communication, and terminal handling capabilities unique to the ARPANET.

We take very seriously the responsibility to provide effective communication capabilities to SUMEX-AIM users and are continuously looking for ways to improve our existing facilities as well as investigate alternatives becoming available. We are investigating the TELENET facilities that have been rapidly expanding this past year. BB&N has hooked one of their TENEX systems up to TELENET and subjective reports are that response problems similar to those reported above were present there as well. We have requested specific data on their experience but have not received any yet. We have received comments particularly from Professor Colby's group which uses the ARPANET primarily (see page 116) that serious network delays place the remote user at a substantial disadvantage in competing for system resources and that compensating biases in allocation procedures should be implemented to offset the problems. Another critical problem is the lack of high speed printing facilities local to remote groups. The new system being installed should help assure remote users their fair share of CPU; but a simple bias in system percentage will not offset network delay problems. The communication problems must be solved as communications problems and the only way to ensure good terminal response is to provide high enough speed lines that are not over loaded.

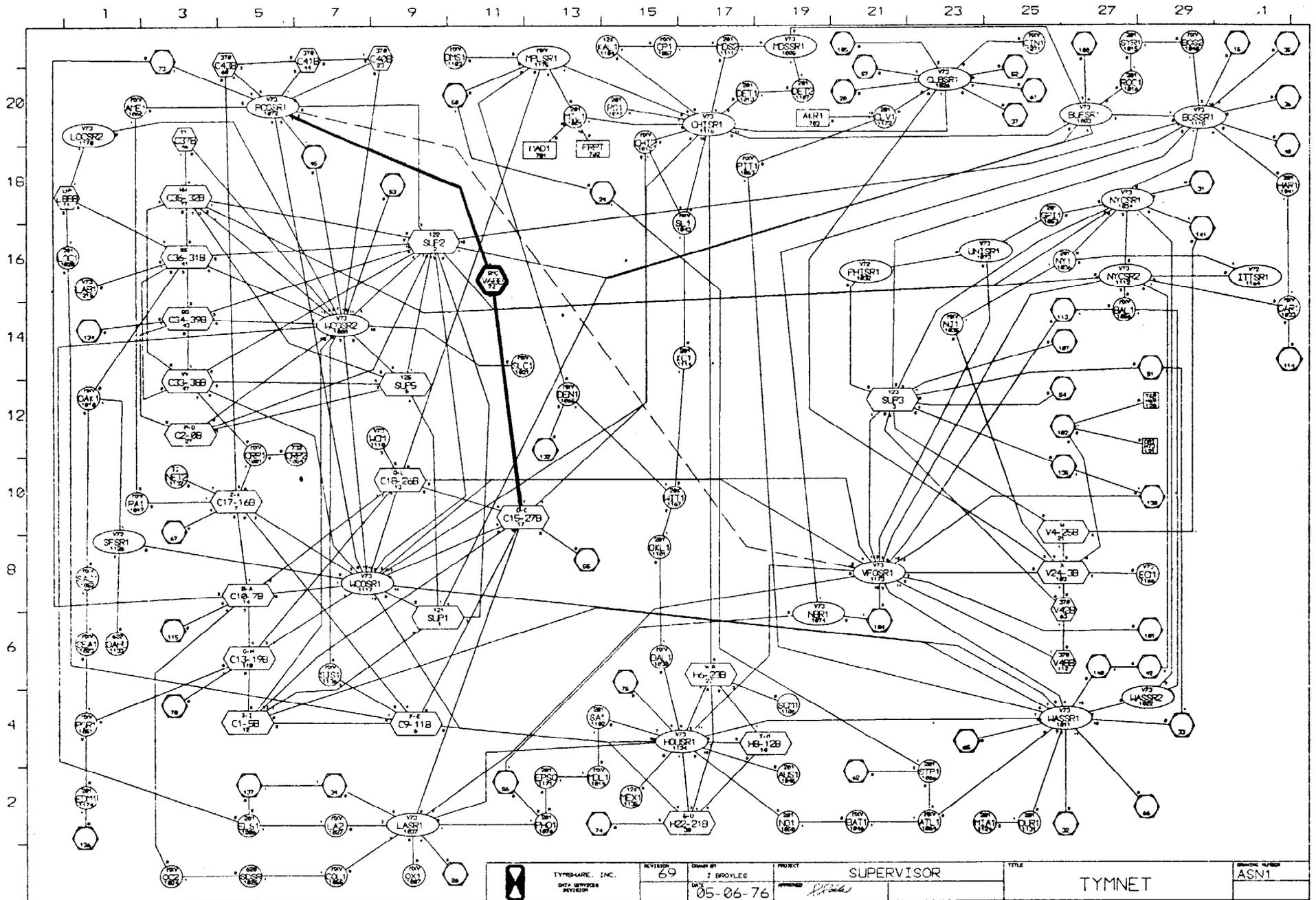


Figure 4. TYMNET Network Map

Figure 5.
TYMNET RESPONSE DELAY STATISTICS

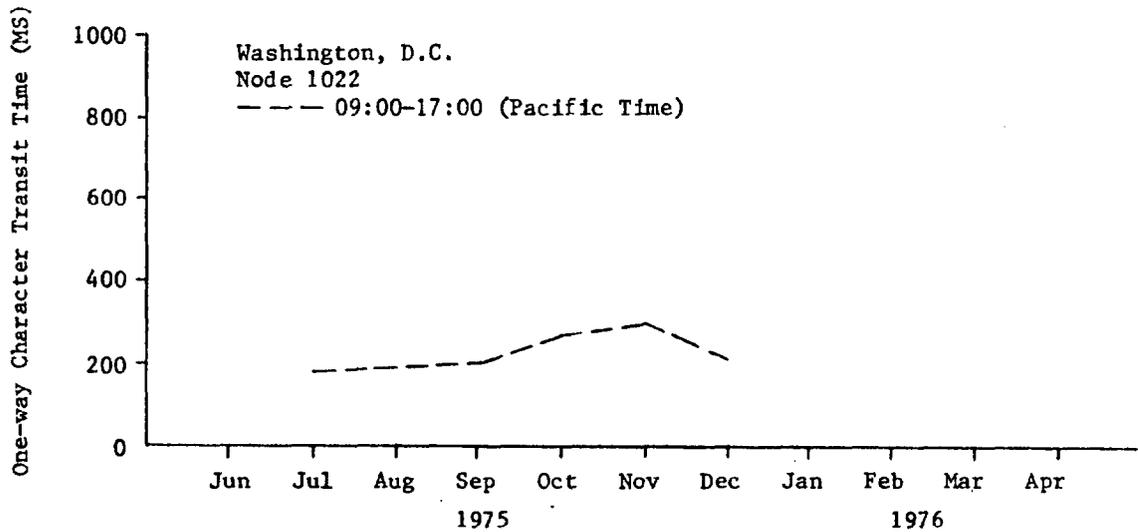
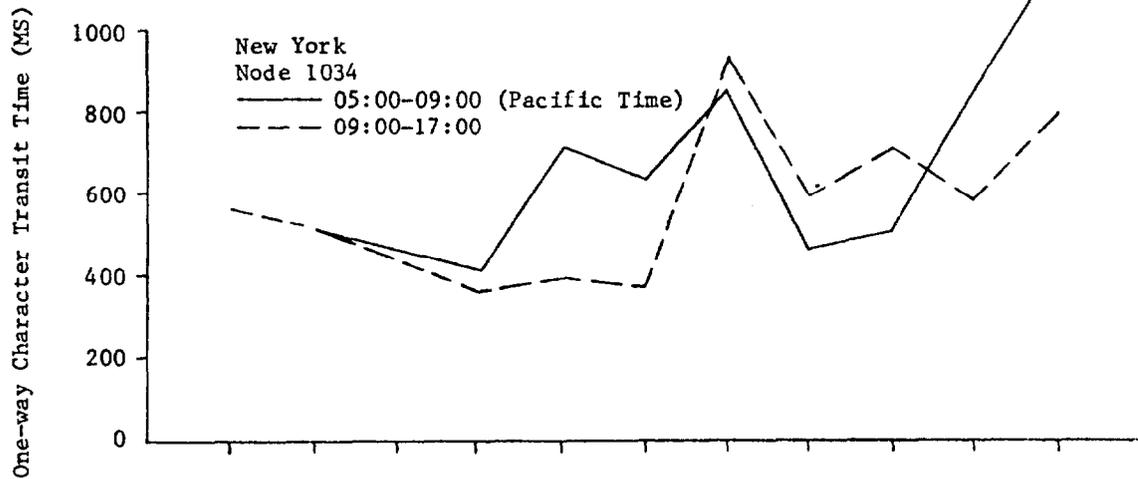
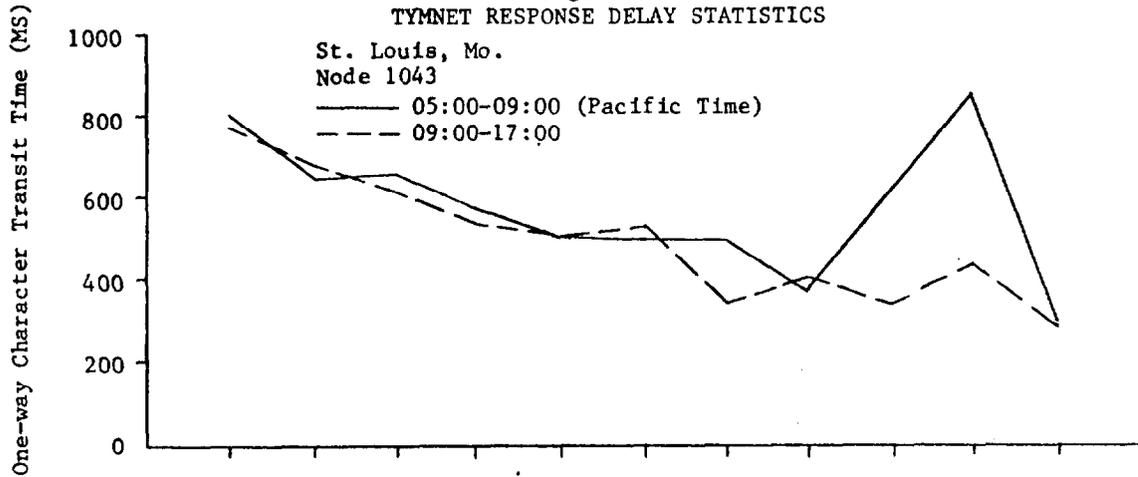


Figure 6.

ARPANET GEOGRAPHIC MAP, FEBRUARY 1976

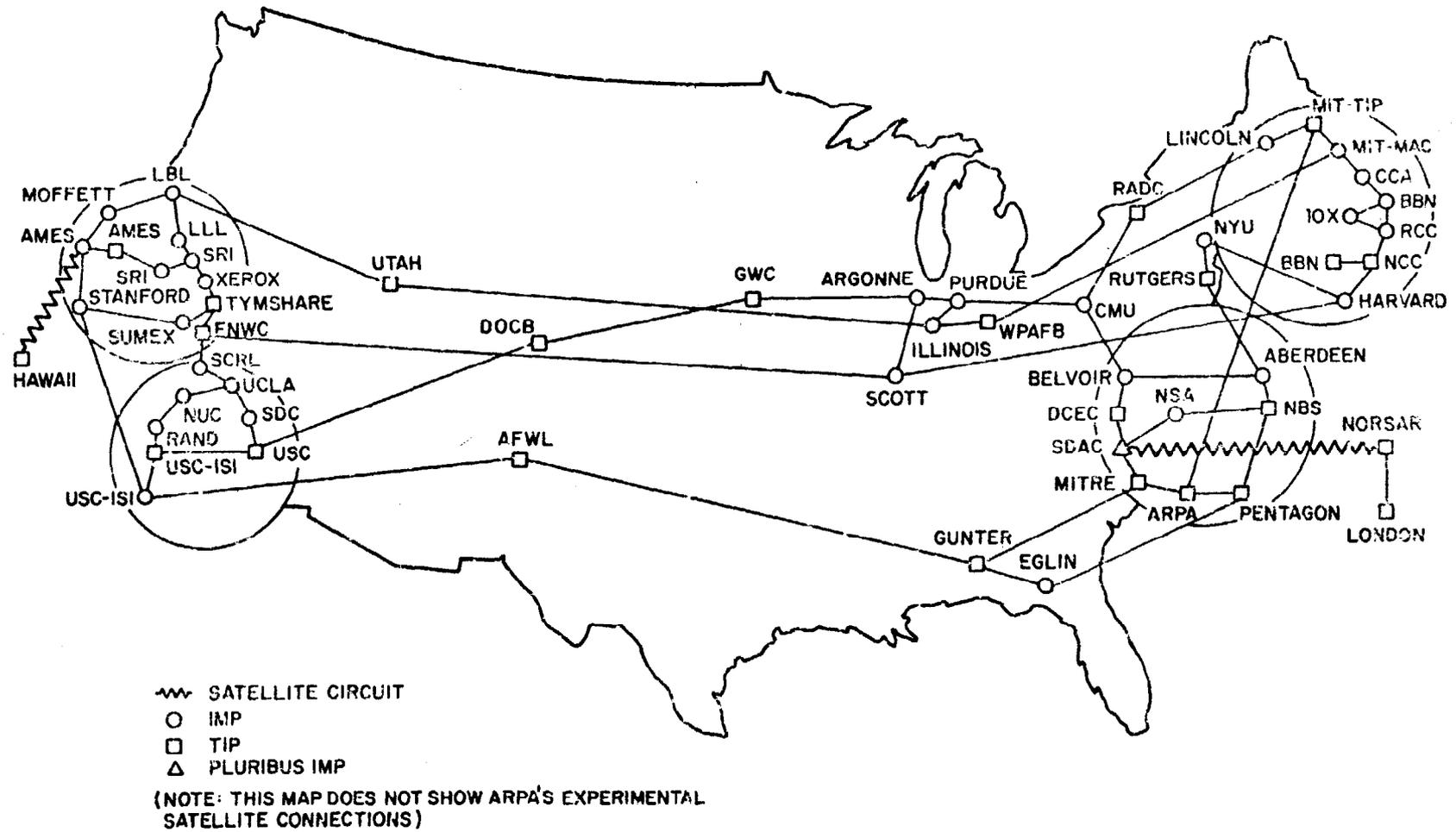
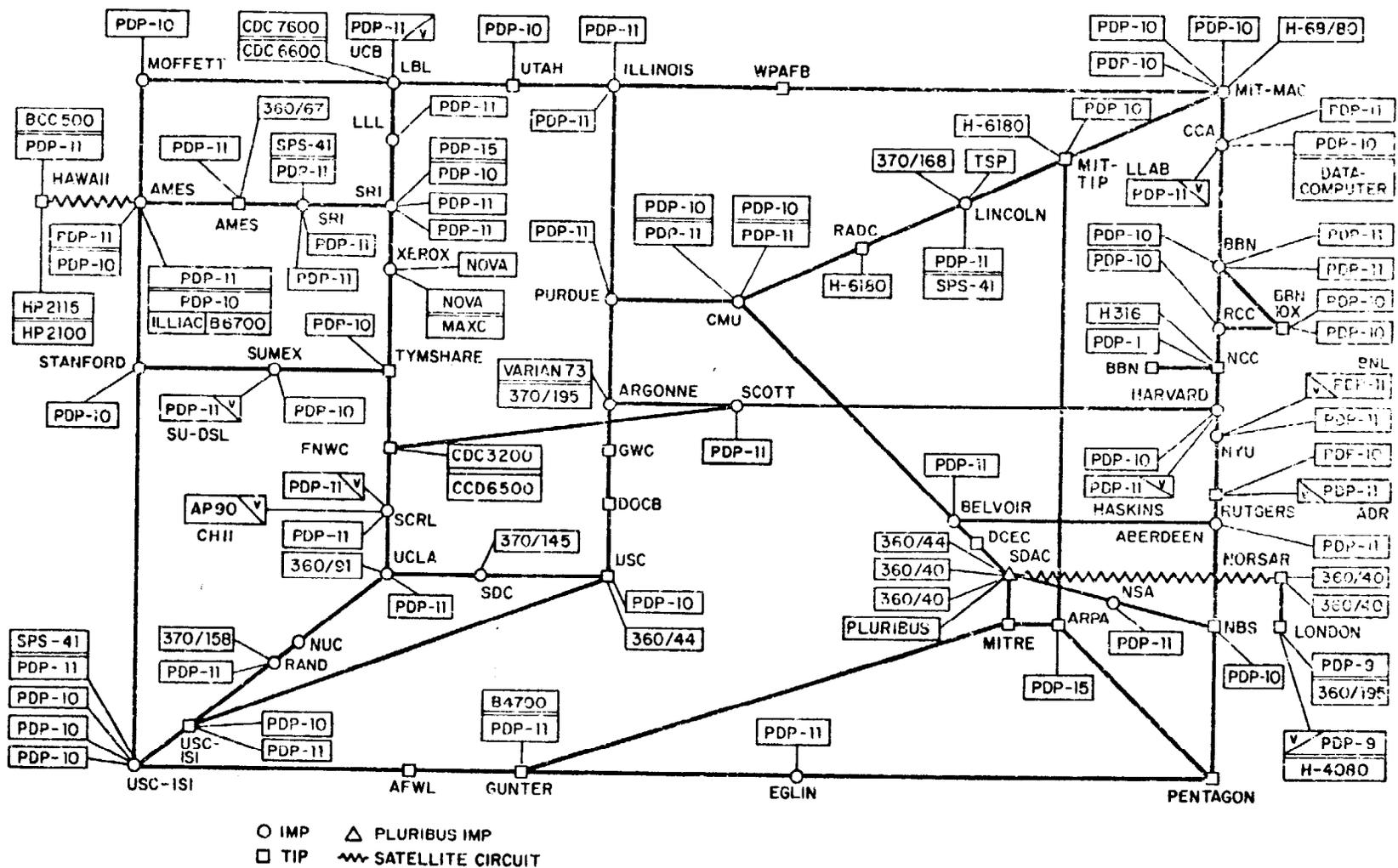


Figure 7.
 ARPANET LOGICAL MAP, FEBRUARY 1976



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

II.A.2.d SYSTEM RELIABILITY AND BACKUP

System reliability has improved over the past year; excellent under stable hardware and software conditions and degrading during debugging and development periods (drum debugging, dual processor work, etc.) and during periods of hardware problems. The pertinent data are given below with indications of periods during which development took place.

SUMEX-AIM CRASH FREQUENCY (crashes/month)
AND DOWN-TIME DATA (hours/month)

Crash Type	1975						1976					
	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR
DEC HARDWARE	10	7	22	26	8	10	4	16	9	6	1	6
SOFTWARE	4	3	6	6	6	0	3	5	1	3	2	5
ENVIRONMENT	0	0	1	0	0	0	1	0	1	1	2	1
TYMNET HDWRE	0	0	0	0	1	0	0	0	0	1	1	0
UNKNOWN	1	0	1	0	0	0	1	0	1	1	0	0
DOWN-TIME												
SCHEDULED	80	79	98	123	72	52	52	43	41	48	38	67
UNSCHED	28	19	30	42	7	4	3	21	26	11	2	7

DEFINITIONS:

Crash = Any occasion on which an operational system must be restarted or reloaded. Multiple crashes while trying to reload are not counted unless the system comes up fully between crashes.

DEC Hardware Crash = Any crash caused by a failure in the PDP-10 hardware or peripheral equipment (CPU, disk, drum, etc.)

Software Crash = Any crash caused by a malfunction within the TENEX software system.

Environmental Crash = Any crash caused by power failure, air conditioning outage, lightning, etc.

TYMNET Hardware Crash = Any crash caused by the TYMNET hardware or the interface to the PDP-10. This includes only the times when a TYMNET problem causes the PDP-10 to crash and not the times when the TYMNET goes down and the PDP-10 continues in operation.

Unknown Crash = All other crashes in which the cause is not assignable.

Scheduled Down-time = Preventive maintenance time (6-8 hours/week), file system backup (3-6 hours/week), scheduled maintenance to repair non-critical component failures, and system development activities requiring a stand-alone machine.

Unscheduled Down-time = Time lost because of unexpected hardware or software failure. For the most part this is the time to diagnose and either repair the problem or to reconfigure the system and bring it up to run in a somewhat degraded mode until a later scheduled shutdown for permanent repair.

Whenever development efforts are undertaken which affect the system hardware or monitor, additional downtime and some period of unreliability may result causing more crashes than are representative of the overall reliability of the system. The following gives some insight into these development efforts as reflected in the above data.

Jul - Sep 1975: Debug drum system error rate problem.

Late Apr 1976: Begin dual processor installation.

As can be seen, we have had some periods of hardware unreliability stemming mostly from intermittent problems. Particularly troublesome components of the system in terms of such problems have been the disk drives, memories, and during hardware relocations, the inter-device cabling. The KI-10 CPU has been very stable and given only one problem over the past year (an I/O bus driver).

From the user's viewpoint, besides the obvious inconvenience of not being able to work during down time, the fragility of the highly interlinked TENEX file system has caused only a few occasions of having to backup to previous file system states this past year. We save changed files daily and copy the entire file system to fresh disk packs weekly. Thus an unexpected crash may cause the loss of up to one day's worth of work - it in fact may take longer for a given user to reconstruct the lost work if complex debugging or development changes were involved and undocumented. When the system is known to be subject to intermittent crashes, we backup more often to protect users.

Our current schedule for system backup is early Sunday morning (Pacific Time). We now have two students who do the file system backups at night as well as the archive/retrieve requests. By moving these activities to night hours, we off-load them from the prime time and also provide added coverage for quick recovery from any system crashes. This does not require full time attention and the students also help out with system programming tasks in developing utilities.

Another aspect of reliability and backup is the need to assure computing service for critical demonstrations, lectures, and the like. We have a good mutual relationship with existing ARPANET TENEX sites for such backup when needed (e.g., for the AIM workshop).

II.A.2.e PROGRAMMING LANGUAGES

Over the past year we or members of the SUMEX-AIM community have continued to maintain the major languages on the system at current release levels, have TENEXized several languages to improve efficiency, and have investigated a number of issues related to the efficiency of programs written in various LISP implementations and the exportability of programs. These issues are becoming increasingly critical in dealing with AI performance programs which have reached a level of maturity so that substantial, non-developmental user communities are growing. The following summarizes general accomplishments and the following section discusses in detail the work this past year in designing a machine-independent SAIL system (MAINSAIL).

General Language Support:

The ALGOL-like modeling language, SIMULA, was requested by the DENDRAL group for consideration as a language in which to implement a more efficient version of the chemical structure generation programs. The most recent release of SIMULA has been brought up on the system. It is also used by a number of the Rutgers project members.

Two existing programming languages were TENEXized by Mr. Tom Wolpert of IMSSS. TNXFAIL is now the official version of FAIL for TENEX sites. His code has been incorporated under compilation switch into the standard FAIL sources maintained at the Stanford Artificial Intelligence Laboratory (SU-AI). Mr. Wolpert also TENEXized UC Irvine - LISP (ILISP) which is an extension of LISP 1.6 to include the break package and editor facilities of INTERLISP circa 1971. ILISP is used extensively by Prof. Colby's group at SUMEX.

The latest DEC release of FORTRAN10 was installed late last year and is relatively stable although several bug fixes have since been made. As part of an effort to remain current with all new DEC releases, we have also updated the versions of: MACRO, BLIS10, and BASIC.

Two other languages which received active maintenance at SUMEX this year are INTERLISP and SAIL. New versions of INTERLISP are continually being issued by XEROX-PARC and are brought up on SUMEX by Mr. Larry Masinter (of Xerox) and Ms. Suzanne Johnson. Because of the large number of LISP programs that are written in various versions of INTERLISP (which are not necessarily compatible with the new releases) and the need to keep