BOOK I

SUMEX-AIM Resource Progress Report

This is an annual report of the work performed under an NIH Biotechnology Resources Program grant supporting the Stanford University Medical EXperimental computer (SUMEX) research resource for applications of Artificial Intelligence in Medicine (AIM). It spans the year from May 1976 - April 1977. As we have invested substantial effort in preparing a related document, an application for renewal dated June 1, 1977, this report has been prepared by revising and augmenting the other. Some sections may inadvertently reflect that provenience, e.g., by adopting a longer time perspective, but we believe without distorting or misrepresenting our last year's effort. Book II of this report is the same text as used for the renewal, and contains detailed progress reports of collaborating user projects and other pertinent appendices.

1 RESOURCE OBJECTIVES AND PROGRESS

1.1 OVERVIEW OF OBJECTIVES AND RATIONALE

The SUMEX-AIM project is a national computer resource with 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.

Definitive funding of the SUMEX-AIM resource was initiated in December 1973. The principal hardware was delivered and accepted in April 1974, and the system became operational for users during the summer of 1974. The present renewal is therefore written from a perspective of just short of three years of experience in attempting to develop and serve the user community for the resource.

The original SUMEX proposal was an outgrowth of two lines of endeavor at Stanford that had been supported by the Biotechnology Resources Program. The ACME project (Advanced Computer for MEdical Research), 1965-72, had introduced the innovation of interactive time-shared computing to the medical research community at the Stanford Medical Center. The second line, the DENDRAL project, is a resource-related project connected with applications of artificial intelligence to problems of molecular characterization by analytical instruments like mass-spectrometry, gas-chromatography, nuclear magnetic resonance, and so on.

In 1972 we applied to NIH for the establishment at Stanford of a next generation computer resource to supplant ACME for applications for which the university-wide facility was inadequate. The DENDRAL project was the central source of this initiative; several others entailing real-time instrumentation as much as AI needs were also specified. During the subsequent 18 months, we entered a phase of protracted review and negotiations with BRP and its advisory groups, from which emerged the policy determination that resources of this scope were best justified if they could be functionally specialized, but geographically generalized. The emerging technology of computer networking opened an opportunity to demonstrate this model in a way that could serve both local and national needs.

Our technical task has been achieved: to collect and implement an effective set of hardware and software tools supporting the development of large and complex AI programs and to facilitate communications and interactions between user groups. In effect, users throughout the country can turn on their own teletype or CRT-display terminals, dial a local number, and logon to SUMEX-AIM with the same ease as if it were located on their own campus -- and have access to a specialized resource unlikely to be matched nearby. From the community viewpoint, we have substantially increased the roster of user projects (from an initial 5) to 11 current major projects plus a group of pilot efforts. Many of these projects are built around the communications network facilities we have assembled; bringing together medical and computer science collaborators from remote institutions and making their research programs available to still other remote users. As discussed in the sections describing the individual projects, a number of the computer programs under development by these groups are maturing into tools increasingly useful to the respective research communities. The demand for production-level use of these programs has surpassed the capacity of the present SUMEX facility and has raised the general issues of how such software systems can be optimized for production environments, exported, and maintained.

1.2 BACKGROUND AND PROGRESS

1.2.1 PROGRESS SUMMARY

This progress summary covers the period from December 1973, when the SUMEX-AIM resource was initially funded, through April 1977. During this period we have met all of the defined goals of the resource:

i) We have established an effective computing facility to support a nation-wide community of medical AI research projects including connections to two computer communication networks to provide wide geographical access to the facility and research programs.

ii) We have actively recruited a growing community of user projects and collaborations. The initial complement of collaborators included five projects. This roster has grown to eleven fully authorized projects currently plus a group of approximately six pilot efforts in various stages of formulation. Recruiting efforts have included a public dedication and announcement of the resource, NIH referrals from computer-based project reviews, direct contacts by resource personnel and on-going projects as well as contacts through the AIM workshop series coordinated by the Rutgers Computers in Biomedicine resource under Dr. Saul Amarel.

iii) We have established an AIM community management structure based on an overseeing Executive Committee and an Advisory Group to assist in recruiting and assessing new project applications and in guiding the priorities for SUMEX-AIM developments and resource allocations. These committees also provide a formal mechanism for user projects to request adjustments in their allocated share of facility resources and to make known their desires for resource developments and priorities.

iv) SUMEX user projects have made good progress in developing more effective consultative computer programs for medical research; one of the major goals toward which our AI applications are aimed. These performance programs provide expertise in analytical biochemical analyses and syntheses, medical diagnoses, and various kinds of cognitive and affective psychological modeling.

v) We have worked hard to build system facilities to enable the inter- and intra- group communications and collaborations upon which SUMEX is based. We have a number of examples in which user projects combine medical and computer science expertise from geographically remote institutions and numerous examples of users from all over the United States and occasionally from Europe experimenting with the developing AI programs. The SUMEX staff itself has had good success in establishing such sharing relationships on a system level with other research groups and has many examples of complementary development and maintenance agreements for system programs.

vi) We have made numerous improvements to the computing resource to extend its capacity, to improve its efficiency, to enhance its human interfaces, to improve its documentation, and to enhance the range of software facilities available to user projects.

vii) We have begun a core research effort to investigate alternatives and programming tools to <u>facilitate the exportability of user and system software</u>. This is just now producing a "machine-independent" implementation of the ALGOL-like SAIL language which will run on a range of large and small machines and provide a language base for transferring programs.

viii) We have <u>supported community efforts</u> in the more systematic <u>documentation of AI concepts and techniques</u> and in <u>building more general software tools</u> for the design and implementation of AI application programs. These have included a Stanford AI Handbook project comprising a compendium of short articles about the projects, ideas, problems, and techniques that make up the field of AI.

1.2.2 DETAILED PROGRESS REPORT

The following material covers in greater detail the SUMEX-AIM resource activities over the past 3.5 years. These sections attempt to define in more detail the technical objectives of our research community and include progress in the context of the resource staff and the resource management. Details of the progress and plans for our external collaborator projects are presented in Section 6 on page 41 (in Book II).

1.2.2.1 DEFINITION OF TERMS AND OBJECTIVES

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). Currently authorized projects in the SUMEX community are concerned in some way with the application of these principles to biomedical research. 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, constrained where appropriate by heuristic rules or interactions with the user, has already produced promising results in areas such as chemical structure elucidation and synthesis, 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.

An equally important function of the SUMEX-AIM resource is an exploration of the use of computer communications as a means for interactions and sharing between geographically remote research groups in the context of medical computer science research. This facet of scientific interaction is becoming increasingly important with the explosion of complex information sources and the regional specialization of groups and facilities that might be shared by remote researchers. 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

---

(1) For recent reviews to give some perspective on the current state of AI, see: (i) Winston, P.H., "Artificial Intelligence", Addison-Wesley Publishing Co., 1977; (ii) Nilsson, N.J., "Artificial Intelligence", Information Processing 74, North-Holland Pub. Co. (1975); and (iii) a summary by Feigenbaum, E. A., attached as Appendix I, page 202 (see Book II). 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 II on page 225 in Book II).

geographically separate institutions; separate both from each other and from the computer resource. The network experiment also enables 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, we have been able to demonstrate that 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. In a similar way, the network connections have made possible close collaborations in the development and maintenance of system software with other facilities.

1.2.2.2 FACILITY HARDWARE

Based on the AI mission of SUMEX-AIM, we selected a Digital Equipment Corporation (DEC) model KI-10 computer system for our facility. This selection was based on 1) hardware architectural and performance features, 2) available software support relevant to AI applications, 3) price versus performance data for the system, and 4) the scope of the user community from which we might expect to draw collaborators and share software. This choice has proved highly effective.

The current system hardware configuration is diagrammed in Figure 1 on page 10. It is the result of a number of augmentations over the past 3 years to meet the capacity needs of the growing SUMEX-AIM project community. Our initial configuration consisted of a KI-10 processor, core memory (192K 36-bit words @ 1 microsecond), swapping storage (1.7M words @ 8 msec average rotational latency and 2 microsecond/word transfer rate), file storage (40M words), magnetic tapes, DEC tapes, terminal line scanner, and line printer. Our network connections are discussed in Section 1.2.2.4 on page 16.

This system reached prime-time saturation by fall of 1974. Since many of our medical and other professional collaborators cannot adjust their schedules to match light computer loading during the night-time hours, the prime-time responsiveness is crucial to being able to support medical experimentation with developing programs and to allow community growth. We have taken active steps to transfer as much prime-time loading as feasible to evening and night hours including shifting personnel schedules (particularly for Stanford-based projects), controlling the allocations of CPU resources between various user communities and projects, and encouraging jobs not requiring intimate user interaction to run during off hours by developing batch job facilities. Despite these efforts, prime-time loading has remained quite high, particularly with the growth of the number of user projects.

A similar congestion has persisted in the on-line file space we have been able to allocate to user projects. Again we have implemented controls to try to assure effective use of available space and to encourage use of external file storage facilities such as the ARPANET Data Computer and other computer sites. Nevertheless, the interactive character of SUMEX use, the large AI program files, and the extensive use of SUMEX for collaborator communications have continuously raised file space demands beyond those we could meet.

We have proposed a number of hardware configuration augmentation steps to the Executive Committee to cost-effectively provide additional capacity. These were based on analyses of predominant system bottlenecks and enhancement steps feasible within available budgets. The enhancements approved by the committee and implemented include:

1) Add 64K words of core memory and 20M words of file storage (11/74)

2) Add second KI-10 CPU for dual processor operation (5/76)

3) Add 256K words of core memory and upgrade file system to higher volume, lower cost technology (recently approved by NIH and the AIM Executive Committee with implementation in progress)

A plot of effective CPU capacity as a function of continuing investment is shown in Figure 2 on page 11 and displays the cost-effectiveness of our sequential augmentations. At the present time our hardware configuration has grown about as much as is cost-effective. Additional growth would entail significant redesigns of the system including upgrades of existing hardware. Contemplating such future expansion also raises the issues of compatibility with newer hardware technologies being announced. These provide advantages in speed, cost, size, and maintainability. Such a complete upgrade is not envisioned in the immediate future as a number of interesting new product announcements are expected over the next 1 or 2 years that could substantially affect such an upgrade strategy.

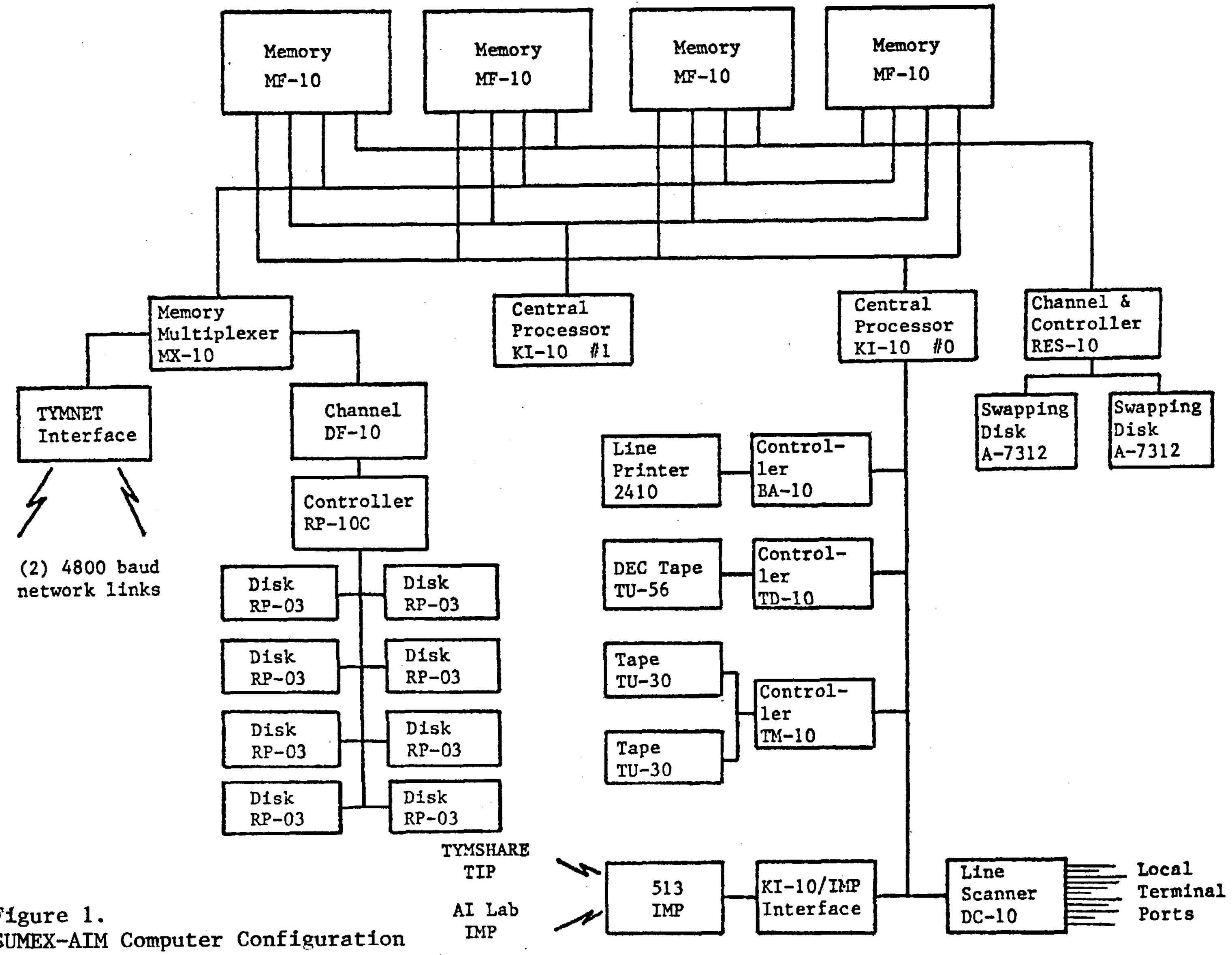
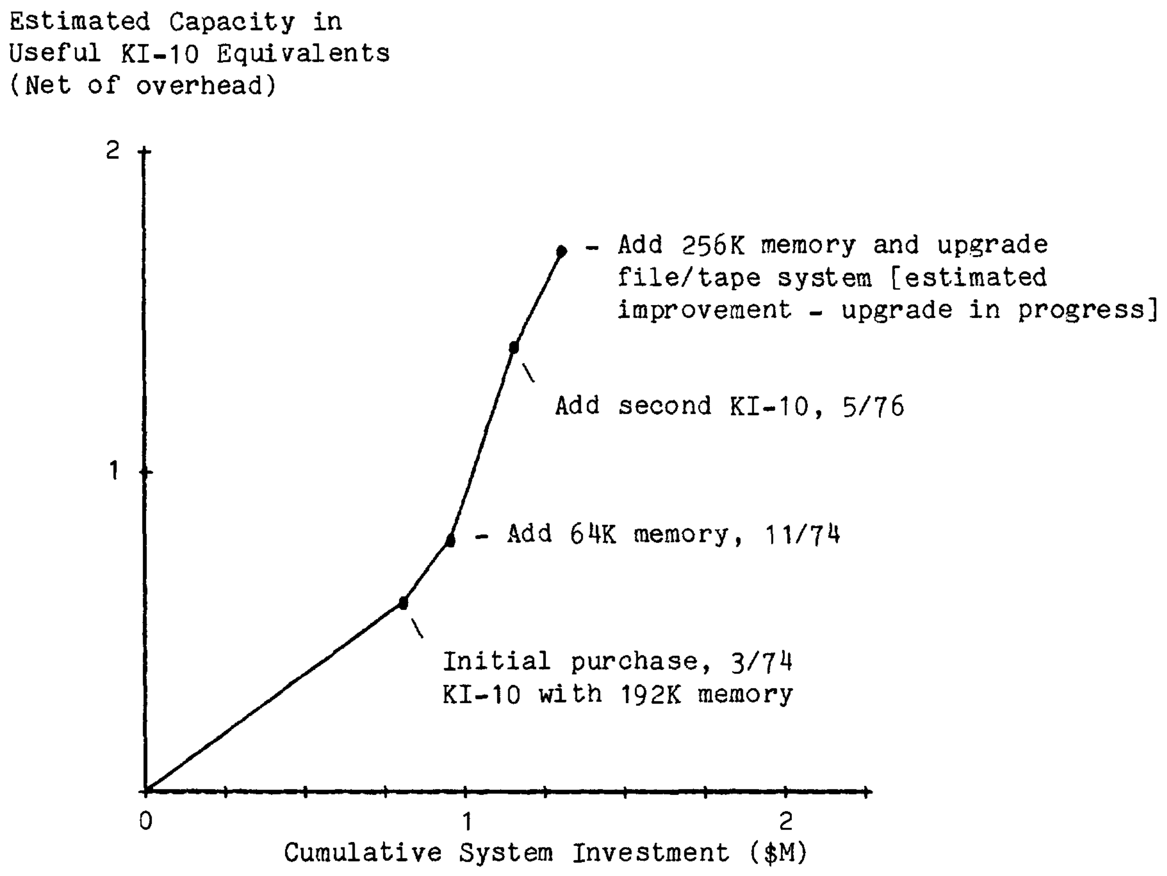


Figure 1.
SUMEX-AIM Computer Configuration

Figure 2. Cost-effectiveness of SUMEX Augmentations



This plot illustrates the incremental increases in computing capacity achieved as a function of cumulative investment in the SUMEX-AIM facility. The higher slope of the curve after the initial investment illustrates both the substantial investment in peripheral devices (file system, tapes, communications, etc.) and the trend toward lower memory prices. The largest impact in terms of PDP-10 memory price reductions occurred around the time of adding the 64K increment in November 1974. Since then processor prices have stayed relatively stable and memory prices have dropped less dramatically. It should be noted that semi-conductor memories have not yet made a big in-road in the PDP-10 market; this technology is where the more recent memory price reductions have occurred.

The original purchase of 1 KI-10 with 192K of memory for about $800K performed with about 60% efficiency under peak load. Adding the 64K of memory for $75K brought the efficiency up to about 85%. Then adding the second processor for $200K increased throughput to about 1.3-1.4 KI-10 equivalents. This step represents about a 59% increase in throughput for a 20% increased investment. A proposal has been approved recently by the AIM Executive Committee and NIH to augment core memory by 256K words. This augmentation would increase throughput to about 1.7 KI-10 equivalents for another $100K; this would be a 26%

throughput increase for 8% additional investment. As part of the proposed memory augmentation we plan to upgrade the file and tape systems as well to relieve file space congestion and increase system operations efficiency. Including the net cost of the file/tape upgrade in these figures (purchase price less resale of existing equipment) raises the proposed additional investment to $160K and the fractional increase from 8% to 13%. Of course, the disk upgrade affects CPU throughput only indirectly in that the increased speed reduces contention, particularly when moving head swapping is necessary. It contributes primarily to supporting the growing on-line file needs of the projects.

Figure 3. Capacity and Loading Increase with Dual Processor Augmentation

	1-PROC OP'N 1/76 - 4/76	2-PROC TRNS'N 5/76 - 8/76	2-PROC OP'N 9/76 - 12/76	2-PROC OP'N 1/77 - 3/77
Peak Ld Ave	4.8	5.6	6.0	6.6
Peak Jobs	30.2	33.3	34.7	38.1
% Overhead/ Processor	18.1	31.1	33.2	31.9
Total CPU Hrs/Mo	304.4	384.9	534.0	520.1

This table presents system usage data averaged over several months preceding, during, and after installation of the SUMEX-AIM dual processor system in order to show real changes in peak loading capacity and computing resources delivered. The first three rows of data are derived from monthly diurnal loading data and reflect average prime-time peak loading conditions (daily peak usage figures are often considerably higher, but those shown better represent gross trends). The last row gives average total monthly CPU hours delivered during the various periods.

With the common criterion that users have pushed both the single and dual processor systems to the limits of useful work in terms of prime time responsiveness, it is clear that the second processor has substantially increased throughput ("tolerable" peak load average up 38%, number of jobs up 26%, and delivered CPU hours up 71%). At the same time the overhead burden per machine has risen from 18 to 32%, principally in the category of I/O wait (total scheduler time and time waiting for a runnable job to be loaded in core). An additional factor, not explicitly shown in these data (because we only have a 1 msec clock), is the added time spent at interrupt level servicing drum swapping. This adds another 10-15% estimated overhead.

We feel these increased overhead figures can be reduced roughly to the single processor levels by adding more memory, thereby effectively recovering about 40-50% of the capacity of a KI-10 processor. A proposal is now pending with the AIM Executive Committee for this augmentation and we expect it to be implemented within the funding ceiling of the current grant.

1.2.2.3 SYSTEM SOFTWARE

In parallel with the choice of DEC PDP-10 hardware for the SUMEX-AIM facility, we selected the TENEX operating system developed by Bolt, Baranek, and Newman (BBN) as the most effective for our medical AI applications work. TENEX was the only available demand-paged system to support simultaneous large address space users, offered the INTERLISP language for LISP-oriented program development, and was well integrated with the ARPANET facilities which provide an excellent base for our community sharing efforts. This choice has proven a very effective one in that the productivity of the TENEX community in AI research has been highly advantageous to us (2).

The original BBN TENEX was written for a hardware-modified KA-10 system. This version of the system required a substantial amount of work to accommodate the relatively limited paging facilities of the KI-10 to run effectively. These early phases also included substantial monitor work to incorporate the TYMNET memory-sharing interface which connects us to the TYMNET and to integrate the high speed swapping storage. We have made numerous enhancements to the monitor calls and corrections of bugs to develop a highly reliable and effective operating system for our community work.

We continue to work to improve the efficiency of the system and its effectiveness in allocating valuable resources. For example we have modified the handling of user page tables so that the expensive procedure of clearing page tables and setting them up to run time-shared users could be minimized. This involved creating a pool of page tables which could be allocated to currently running users and could be kept available without setup overhead. We also implemented a system for migrating dormant pages from our fast swapping storage to moving head disk. This preserves the use of this limited resource for the currently active jobs.

We have implemented a form of "soft" CPU allocation control in the monitor, assisted by a program which adjusts user percentages for the scheduler based on the dynamic loading of the system. The allocation control structure works based on the scheduler queue system and takes account of the _a priori_ allocation of CPU time and that actually consumed. Our TENEX uses a hierarchy of five queues for jobs ranging from highly interactive jobs requiring only small amounts of CPU time between waits to more CPU intensive jobs which can run for long periods without user interaction. These interactive queues (text editting, etc.) are scheduled at highest priority without consideration of allocation percentages. If nothing is runnable from the high priority queues, the CPU-bound queues are scanned and jobs are selected for running based on how much of their allocated time has been received during a given allocation cycle time (currently 100 seconds). If no such jobs are runnable, then those that have received their allocation of CPU time already are scheduled based on how much they are over allocation and how long they have waited to be run again. This system is not a reservation system in that it does not guarantee a given user some percentage of

---

(2) It should be noted that DEC has recently adopted a form of TENEX (TOPS-20) as their choice for future system marketing. They have made improvements in a number of areas of the monitor and subsystem software but have also shown an increasing tendency to make changes to the TOPS-20 system that impair compatibility with older TENEX systems.

the system. It allocates cycles preferentially, trading off a priori allocations with actual demand but does not waste cycles. This allocation control system is still in an experimental state and we are attempting to evolve the "best" policies with the AIM Executive Committee for dividing the system fairly and effectively among the various communities of users.

During the spring of 1976 we implemented a dual processor version of TENEX as the most cost-effective way to increase our processing capacity. In order to upgrade to the new KL-"n" technology, we would have had to replace most of the equipment that had been purchased initially. For the cost of an additional processor and 8 man-months of intensive software development we were able to increase our CPU capacity by 75%. We have an additional 40% equivalent of a KI-10 processor which can be made available by increasing memory to reduce our swapping contention. The dual processor system that has evolved is running quite reliably. It treats the two machines in an almost symmetric manner. The only difference is that one of the machines has all of the I/O equipment attached to it. They both schedule jobs independently and share the rest of the non-I/O-device monitor code. The areas of the monitor involving the management of resources and jobs which cannot be manipulated by both machines simultaneously are protected by a system of locks. We have made some measurements indicating that overhead for lock waits is less than 10%. The overall increase in capacity provided by the processor upgrade is illustrated in Figure 3 on page 13 which measures key loading parameters in the periods before and after the dual processor installation. Observing the delivery of DEC's high-performance KL-TENEX systems over the past 6 months, it seems clear that for the investment, we made the best choice for the community by implementing the dual processor upgrade. We hope to augment the memory soon to finish exploiting the capacity this extra machine provides and to remove some non-linearities remaining in system swapping performance.

Now that the dual processor system has stabilized, we are undertaking another assessment of system performance to be sure we have removed residual and correctable inefficiencies. This study is on-going now.

Finally, over the past year we made several substantial improvements in the "GTJFN" monitor call which interactively acquires handles on file names specified by the user. These extensions allow for more general "wild card" specifications and interactive help in deciding between and searching for existing file name alternatives. They also give the user much more flexibility in designating groups of files and therefore in structuring his data.

With a working dual processor system, the current implementation of allocation controls in our system, the diverging path of the DEC TOPS-20 system, the termination of active BBN TENEX development, and the unique complications of the KI-10 paging system, we have not made any concerted effort to upgrade our TENEX system to the latest BBN release (1.34). The advantages of such an upgrade are not overwhelming in face of the complicated conversion (KI paging, dual processor, special swapping device handler, TYMNET service routines, local JSYS's, etc.) and resulting system unreliability for some period.

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. This work improves system accommodation to users and provides more convenient and useful information about system and job status. Through such features as login default files, directed file search path commands, mail notification, help facilities, better file archival and retrieval commands, and flexible status information, we have tried to make it easier for users to work on the SUMEX-AIM machine.

1.2.2.4 NETWORK COMMUNICATION FACILITIES

A highly important 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. Over the past year we have been substantially aided in exporting the MAINSAIL system through our network connections. Because of the developmental nature of the language at present, it is important that we have close interactions with the user community and that we be able to effectively perform bug fixes and upgrades. Since MAINSAIL by its nature involves operations on a variety of machines and since our access to example systems cannot be entirely local, the network connections to Rutgers, the Stanford AI Lab, and Stanford Research Institute have been invaluable. It would be considerably more difficult to export MAINSAIL and communicate with users via tapes and mail.

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.

Users asked to accept a remote computer as if it were next door will use a local telephone call to the computer as a standard of comparison. 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 relies upon shared community use of telephone lines to procure widespread geographical coverage at substantially reduced cost. However, unless enough total line capacity is provided to meet peak loads, substantial queueing and traffic jams result in the loss of terminal responsiveness.

TYMNET:

Networks such as TYMNET are a complex interconnection of nodes and lines spanning the country (see Figure 4 on page 20). 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 higly 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. Based on delay time statistics collected during the previous year from our TYMSTAT program, the response times are scarcely acceptable. When delay times exceed 200-300 milliseconds, the character printing lag problems become noticable with a full duplex, 30 char/sec terminal. In the past these times have been particularly bad in New York with peak delays approaching 3 seconds one way! Other nodes have shown uniformly high readings as well. These data were reflected in the subjective, but strongly articulated, comments of many of our user groups.

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. TYMNET has also made some upgrades in the internal connectivity and speeds with which data is switched within their node clusters. These changes seem to have had some beneficial effects in that delay times have improved and user complaints have subsided.

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. TYMNET has implemented 1200 baud service in 7 major cities over the past year. Unfortunately many of our users are not in these cities so we have only limited experience with the 1200 baud support.

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 5 and Figure 6 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.

Consistent with the agreements with ARPA when we were granted network access initially, 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 restricted 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 have done preliminary investigations of the TELENET facilities that have been rapidly expanding this past year. BB&N has hooked one of their TENEX systems up to TELENET and whereas we did not have the same quantitative tools we have for measuring response on the TYMNET, we observed TELENET delays at least as long as those encountered on TYMNET. We did the reverse experiment by using long distance telephone to connect from the TELENET node in Washington, D.C. to the SUMEX machine in California and observed the same sort of delays reaching several seconds per character. The TELENET has many attractive feature in terms of a symmetry analogous to that of the ARPANET for terminal traffic and file transfers and being commercial would not have the access restrictions of the ARPANET. However, until the network throughput improves we would not get substantial benefits from connecting to it.

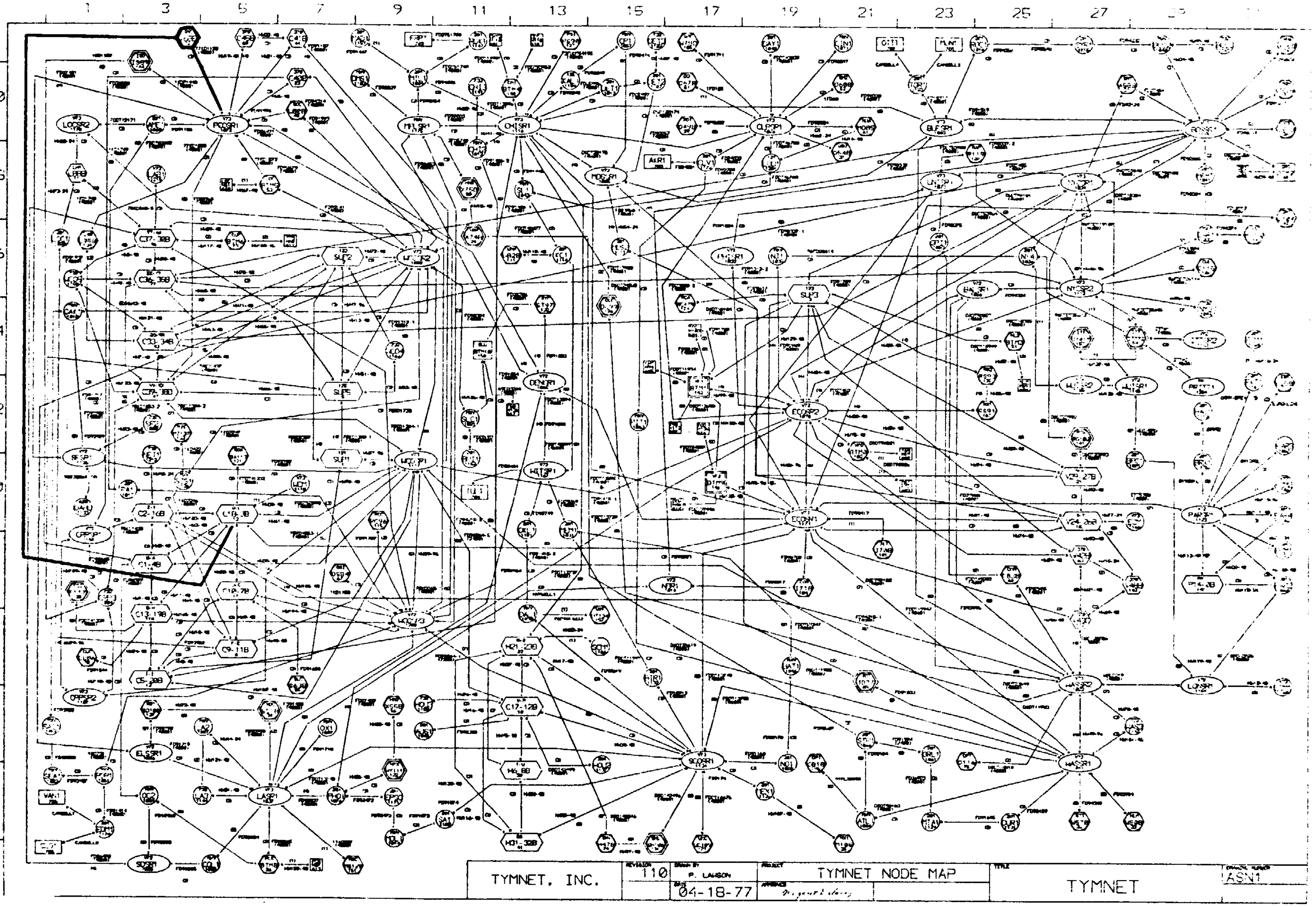
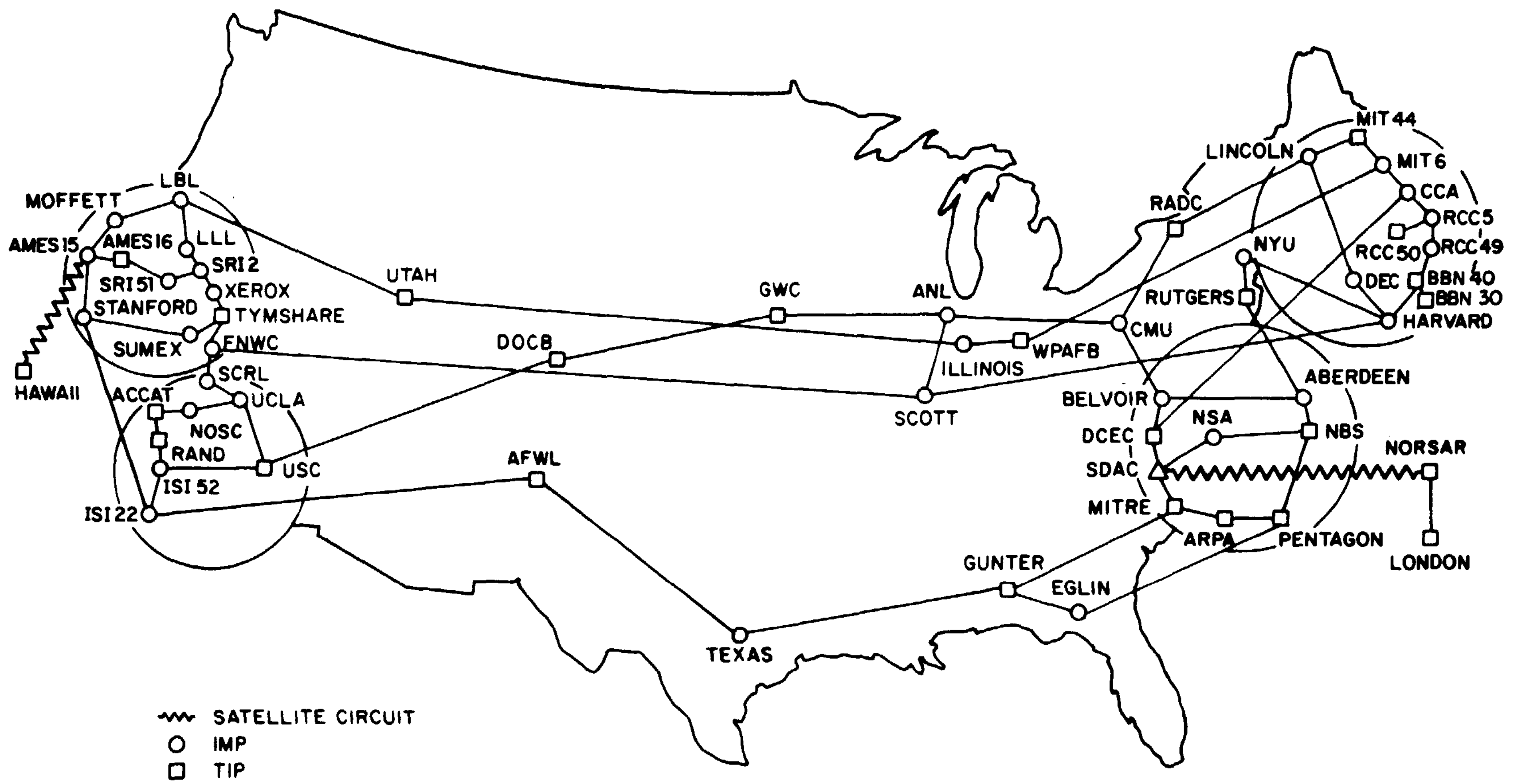


Figure 4. TYMNET Network Map

Figure 5.

ARPANET GEOGRAPHIC MAP, APRIL 1977



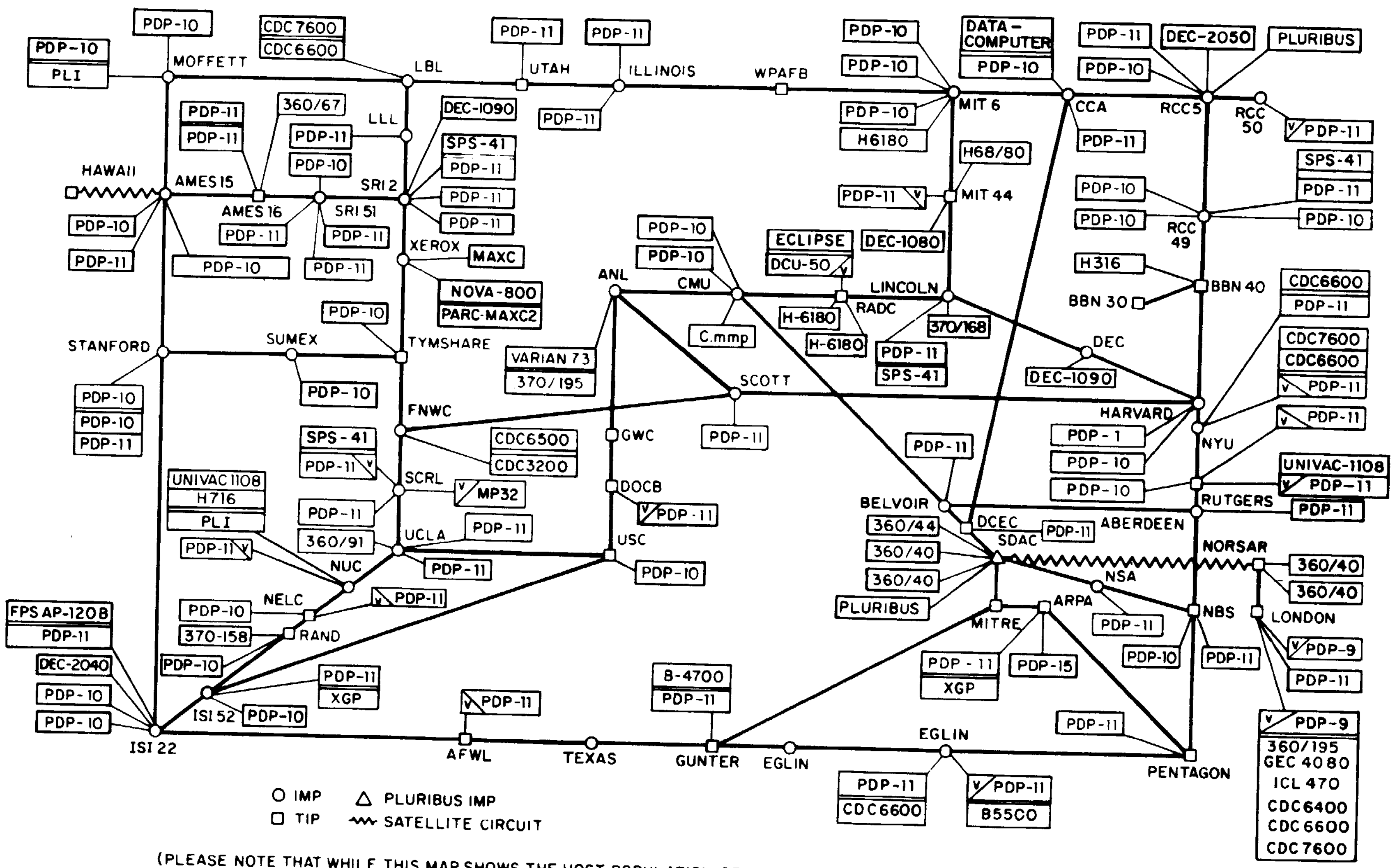
-ww- SATELLITE CIRCUIT
○ IMP
□ TIP
△ PLURIBUS IMP

(NOTE: THIS MAP DOES NOT SHOW ARPA'S EXPERIMENTAL SATELLITE CONNECTIONS)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

Figure 6.

ARPANET LOGICAL MAP, MARCH 1977



(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)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

1.2.2.5 SYSTEM RELIABILITY AND BACKUP

System reliability has remained high over the past years; excellent under stable hardware and software conditions and degrading temporarily during debugging and development periods and during periods of difficult hardware problems. In general we take the system down for approximately 50 hours per month for scheduled hardware maintenance, file backup, and other maintenance. In addition we average from 10 to 15 hours per month in unscheduled downtime. During particularly difficult hardware or software difficulties we must absorb substantially more downtime.

1.2.2.6 PROGRAMMING LANGUAGES

Over the past years 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 ALGOL-like system (MAINSAIL).

LISP Efficiency:

There has been an on-going debate among a number of projects over the best language to choose for developmental implementation of the various AI programs. The key issues include ease and flexibility of conceptual representation of program functions and objects, interactive debugging support, efficiency, and exportability. To date the predominant language choice for AIM research has been LISP and more particularly INTERLISP. These issues are important because they influence the time required to develop new AI programs and subsequently the incremental load placed on the SUMEX machine when in use. We recently attempted an evaluation of INTERLISP and ILISP including the relative efficiencies of the two languages and the level of assistance the language systems provide the user in developing programs. The tests were based on an implementation of a subset of REDUCE (a symbolic algebra manipulator). The results of several iterations in program refinement by experts in the respective languages were that the runtimes for the two versions were quite comparable (far less than the factor of 5-10 disparity predicted by ILISP enthusiasts). A more disquieting result was the substantial difference in runtimes depending on how particular functions were coded IN THE SAME LANGUAGE. It is apparent from the results that factors of 10 differences in time can result from a superficial implementation - expert programming insight is essential to efficient program performance. This is not a real surprise in that it is true of programming in any language - the problems may be increased by such a rich language as INTERLISP with such a wide array of ways to do the same thing but with little guidance as to the relative costs. It has proven very difficult to quantify the "rules" for good programming. Mr. Masinter and Mr. Phil Jackson attempted to document good INTERLISP programming habits and issued a bulletin for SUMEX users.

A further impact of these data is that it is very difficult to simultaneously develop a new AI program and make the implementation highly efficient. With the iterations required to develop the conceptual design of the program, it is difficult to ensure its efficiency. This may lead to the need to reimplement the program after the basic development stabilizes to increase efficiency while still accommodating convenient and orderly further development. Such reimplementation may or may not be best done in LISP - this will depend on many factors including the nature of the program data structure requirements and anticipated further development efforts.

MAINSAIL Progress

SUMEX, in its role as a nationally shared computer resource, is an appropriate vehicle for the development of software unbound by the underlying machine environment. We have a built-in community of program developers acutely aware of the significance of providing their work to a broader base of users. This intersection of hardware capability, software expertise, and dedication to resource sharing presents a unique opportunity to promote a system designed for program sharing.

The MAINSAIL (3) project has three closely related goals:

1) Provide an integrated set of tools for the creation of efficient portable software on a variety of computer systems, and provide support and continued development of these tools in a form compatible across all implementations.

2) Study innovative approaches to portability, both hardware and software, and develop such approaches into effective tools.

3) Promote the development and distribution of portable software, advise and assist in its design, and evaluate its applicability.

By portable software we mean computer programs which may be executed on a variety of machines with few, if any, alterations. MAINSAIL itself will provide the initial example of portable software, since all of the system is written in the MAINSAIL language except for those parts which are determined by the host environment (hardware, instruction set, operating system, etc.). Even these parts are embedded within MAINSAIL.

---

(3) The MAINSAIL (MAchine-INdependent SAIL) language is derived from SAIL, a programming language developed at Stanford University's Artificial Intelligence Laboratory. It is not compatible with SAIL, since SAIL was designed for a PDP-10 with TOPS-10, and hence contains machine-dependencies. However it has retained the basic attributes of SAIL as an extended ALGOL-like language. A summary of some of the features of the MAINSAIL language and their relationship to other languages is given in Appendix III on page 231 (see Book II).

There is a key distinction between MAINSAIL's approach to portability and the "classical" approach characterized by languages such as FORTRAN, ALGOL, LISP, COBOL and BASIC. These languages attempt to adhere to a single syntax standard which is <u>separately implemented</u> for each different computer system. Invariably these implementations have differences which preclude the creation of a program which is accepted by all. It is difficult, if not impossible, to define a language standard which is unambiguous and at the same time sufficiently comprehensible to provide the basis for compatible implementations. Furthermore, many implementors yield to the temptation to provide "enhancements" to the standard which immediately introduces machine and system dependencies.

MAINSAIL, on the other hand, provides a <u>single system</u> (written primarily in itself) which is employed at every site. This is made possible by its ability to compile itself into code for a variety of machines. Only the compiler's code generators and the runtime operating-system interfaces need be rewritten for each implementation. These parts of MAINSAIL are at a level which has already been defined by the machine-independent parts, and do not affect the language from the user's viewpoint. Thus the "language standard" has been reduced to a "semantic standard" which is surrounded by machine-independent software.

It remains to be seen whether the temptation to augment the language with machine-dependencies (for purposes of ultimate efficiency or to take advantage of particular local system features) can be overcome. Herein also lies the biggest "price" to be paid for exportability. The code emitted from the MAINSAIL compiler can be (and is, based on tests to date) at least as efficient as that from many machine-dependent compilers. On the other hand, special machine or operating system features that cannot be uniformly implemented may provide local optimizations at the cost of exportability or vice versa. We cannot effectively measure the extent of this cost at this stage.

DEVELOPMENT APPROACH

We do not underestimate the difficulty in obtaining the cooperation of a community which will span a wide variety of applications and hardware/software systems. If MAINSAIL is to obtain widespread use, it is crucial that it have an effective and credible base of support. The initial parts of MAINSAIL are just about ready for limited distribution. We want to maintain close supervision of this distribution, and insure that systems labelled as MAINSAIL are not altered without our approval. In this regard we are pursuing legal channels to safeguard the integrity of MAINSAIL software. We plan to take MAINSAIL through an orderly progression of development, and to avoid casual distribution with no provision for a solid base of maintenance and future growth.

REVIEW OF PROGRESS TO DATE

MAINSAIL has been under development for almost three years now. Beginning with an initial goal of converting the PDP-10 SAIL compiler to generate code for a PDP-11, several versions had been implemented on a PDP-10 and a PDP-11, and the groundwork had been laid for extending the system to a wider variety of machines. The current version was begun in August of 1976.