NATIONAL INSTITUTES OF HEALTH
DIVISION OF RESEARCH RESOURCES
BIOTECHNOLOGY RESOURCES PROGRAM

SECTION I - RESOURCE IDENTIFICATION

Report Period:

From August 1, 1977 to July 31, 1978
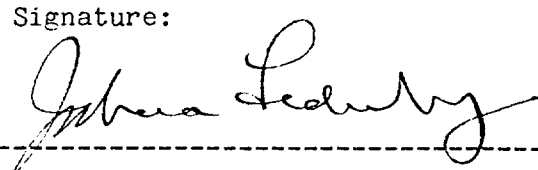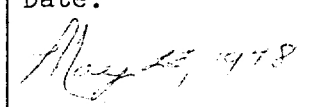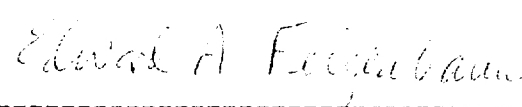
Grant Number:
RR-00785

Report Prepared:
May, 1978

Name of Resource: Stanford University Medical EXperimental Computer (SUMEX)	Resource Address: Stanford University Stanford, California 94305	Resource Telephone Number: (415) 497-5141
Principal Investigators: Joshua Lederberg, Ph.D. Edward A. Feigenbaum, Ph.D.	Titles: Chairman and Professor Chairman and Professor	Academic Departments: Department of Genetics School of Medicine Department of Computer Science
Grantee Institution: Stanford University	Type of Institution: Private University	Investigators' Telephone Nos.: Lederberg: (415) 497-5801 Feigenbaum: (415) 497-4079

Name of Institution's Biotechnology Resource Advisory Committee:
SUMEX-AIM Executive Committee

Membership of Biotechnology Resource Advisory Committee:

NAME	TITLE	DEPARTMENT	INSTITUTION
Saul Amarel, Ph.D.	Chairman & Professor	Computer Science	Rutgers University
Stanley Cohen, M.D.	Head Professor	Div. Clinical Pharmacology Medicine, Genetics	Stanford University School of Medicine
Donald Lindberg, M.D.	Professor Director Director	Pathology Information Science Group Health Care Technology Ctr.	University of Missouri School of Medicine University of Missouri - Columbia
Jack Myers, M.D.	University Professor of Medicine	At Large	University of Pittsburgh School of Medicine

Principal Investigators:	Signature:	Date:
Joshua Lederberg, Ph.D. Chairman and Professor	*Joshua Lederberg*	*May 24, 1978*
Edward A. Feigenbaum, Ph.D. Chairman and Professor	*Edward A Feigenbaum*	*May 23, 1978*
Stanford University Official: Janet P. Jonnson Sponsored Projects Officer	*Janet P. Jonnson*	*5/24/78*

SUMEX-AIM Resource Progress Report - Year 05

This annual report covers work performed under NIH Biotechnology Resources Program grant RR-785 supporting the Stanford University Medical EXperimental computer (SUMEX) research resource for applications of Artificial Intelligence in Medicine (AIM). It spans the year from May 1977 - April 1978.

2 RESOURCE OPERATIONS

2.1 PROGRESS

2.1.1 RESOURCE SUMMARY AND GOALS

The SUMEX-AIM project is a national computer resource with a dual mission: a) the promotion of applications of artificial intelligence (AI) computer science research to biological and medical problems and b) the demonstration of computer resource sharing within a national community of health research projects. The SUMEX-AIM resource is located physically and administratively in the Stanford University Medical School and serves as a nucleus for a community of medical AI projects at universities around the country. SUMEX provides computing facilities tuned to the needs of AI research and communication tools to facilitate remote access, inter- and intra-group contacts, and the demonstration of developing computer programs to biomedical research collaborators.

Artificial Intelligence research is that part of Computer Science concerned with the symbol manipulation processes that produce intelligent action (1). By "intelligent action" is meant an act or decision that is goal-oriented, is arrived at by an understandable chain of symbolic analysis and reasoning steps, and utilizes knowledge of the world to inform and guide the reasoning.

Some scientists view the performance of complex symbolic reasoning acts by computer programs as the sine qua non for artificial intelligence programs, but this is necessarily a limited view.

---

(1) For recent reviews to give some perspective on the current state of AI, see: (i) Boden, M., "Artificial Intelligence and Natural Man," Basic Books, New York, 1977; (ii) Feigenbaum, E.A., "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering," Proceedings of the Fifth International Conference on Artificial Intelligence, 1977; (iii) Winston, P.H., "Artificial Intelligence", Addison-Wesley Publishing Co., 1977; and (iv) Nilsson, N.J., "Artificial Intelligence", Information Processing 74, North-Holland Pub. Co. (1975). An additional overview of research areas and techniques in AI is being developed as an "Artificial Intelligence Handbook" under Professor E. A. Feigenbaum by computer science students at Stanford (see page 123 for a status report and Appendix I for a current outline).

Another view unifies AI research with the rest of computer science. It is a simplification, but worthy of consideration. The potential uses of computers by people to accomplish tasks can be "one-dimensionalized" into a spectrum representing the nature of the instructions that must be given the computer to do its job; call it the WHAT-TO-HOW spectrum. At the HOW extreme of the spectrum, the user supplies his intelligence to instruct the machine precisely HOW to do his job, step-by-step. Progress in computer science may be seen as steps away from that extreme "HOW" point on the spectrum: the familiar panoply of assembly languages, subroutine libraries, compilers, extensible languages, etc. illustrate this trend.

At the other extreme of the spectrum, the user describes WHAT he wishes the computer, as his instrument, to do for him to solve a problem. He wants to communicate WHAT is to be done without having to lay out in detail all necessary subgoals for adequate performance yet with a reasonable assurance that he is addressing an intelligent agent that is using knowledge of his world to understand his intent, complain or fill in his vagueness, make specific his abstractions, correct his errors, discover appropriate subgoals, and ultimately translate WHAT he wants done into detailed processing steps that define HOW it shall be done by a real computer. The user wants to provide this specification of WHAT to do in a language that is comfortable to him and the problem domain (perhaps English) and via communication modes that are convenient for him (including perhaps speech or pictures).

The research activity aimed at creating computer programs that act as "intelligent agents" near the WHAT end of the WHAT-TO-HOW spectrum can be viewed as the long-range goal of AI research. Historically, AI research has been the primary vehicle for progress toward this objective, although a substantial part of the applied side of computer R&D has related goals, if an often fragmented approach. Unfortunately, workers in other scientific disciplines are generally unaware of the role, the goals, and the progress in AI research. Currently authorized projects in the SUMEX community are concerned in some way with the design of "intelligent agents" applied 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 systematic search potential of computerized hypothesis formation and knowledge base utilization, constrained where appropriate by heuristic rules, empirical data, 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 engaged in biomedical computer

3   J. Lederberg & E. Feigenbaum

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 (see Appendix II on page 223). 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. The network experiment also
enables diverse projects to interact more directly and to facilitate selective
demonstrations of available programs to physicians, scientists, and students.
Even in their current developing state, communication facilities enable effective
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). In
a similar way, the network connections have made possible close collaborations in
the development and maintenance of system software with other facilities.

As we complete the first 5-year term of the SUMEX-AIM resource grant, we
can report that our initial technical task has been achieved. We have collected
and implemented an effective set of hardware and software tools to support the
development of large, complex AI programs and to facilitate communications and
interactions between user groups. We have substantially increased the roster of
user projects (from an initial 5) to 15 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.

A number of significant events and accomplishments affecting the SUMEX-AIM
resource occurred during the past year:

1) Professor Lederberg has been the principal investigator and chairman of
   the SUMEX-AIM Executive Committee during the past 5 years. He has now
   been named president of Rockefeller University, effective July 1, 1978.
   He will be succeeded as SUMEX principal investigator by Professor Edward
   Feigenbaum, who is chairman of the Stanford Computer Science Department
   and has been closely associated with the resource since its inception.
   The coordination of project activities with medical research is the
   responsibility of Professor Stanley Cohen, Dr. Lederberg's successor as
   chairman of the Department of Genetics in the Stanford Medical School.
   Professor Lederberg will maintain close ties with these activities as
   chairman of the SUMEX-AIM Executive Committee and through his plans to
   encourage AI applications work at Rockefeller.

2) The SUMEX renewal application submitted last year at this time has been
   reviewed and approved by the National Advisory Research Resources Council.

Our proposed renewal term of 5 years was reduced to 3 years in view of the management changes in progress.

3) We have made a number of upgrades to the SUMEX facility hardware and software systems to enhance throughput and to better control the allocation of resources. We are also establishing a connection to the commercial TELENET network to explore more cost-effective ways to meet community communications needs.

4) We have made progress in the investigation of alternative schemes for the export of programs. A demonstration of the machine-independent MAINSAIL system is nearing completion for the initial set of target machines. The DEC 2020 system, formally announced early this year, provides a relatively inexpensive software-compatible machine for export or expansion of computing capacity for small research groups.

5) The progress of SUMEX-AIM user projects in the development of their respective programs is reported by the individual investigators. We have worked hard to meet their needs and are grateful for their expressed appreciation.

Valediction - Personal remarks by J. Lederberg

While Ed Feigenbaum and I cheerfully accept the full responsibility that is
entailed by our roles as co-investigators of this resource project, we are
embarrassingly aware how much of the effort has been the work of others. Choices
for praise are always invidious, but I have no difficulty in singling out Tom
Rindfleisch as the one person who deserves the most particular credit for the
success of this program. His technical insight and finesse in the system design
and implementation, and in the management of the resource staff are measured by
the visible efficiencies and clarity of documentation of the resource. He is
also preeminently responsible for the drafting of these reports and for managing
our fortunes through all the complexities of federal and university
accountability, and our obligations to local and national users.

Tom would be the first to insist on acknowledging the dedicated support of
the administrative, programming and engineering staffs: I mention Carole Miller
and Karen Carpenter, Rainer Schulz and Andy Sweer, and Nick Veizades as
representatives of the several groups of veterans who have been part of SUMEX-AIM
from its inception, and of the most conscientious team of my experience.

For my own role, I have leaned heavily on my friend and associate, Ed
Feigenbaum, and it is gratifying to be so confident that the work we started
together in building SUMEX-AIM will continue under his able stewardship. Elliott
Levinthal and Bruce Buchanan did a great deal to make all this possible, and to
make the tasks that Ed and I will have taken on not just manageable but fun.
Carl Djerassi, in chemistry, was an indispensable fomenter of the scientific
collaborations. Stan Cohen is making an equally great contribution, both by
succeeding me as chairman of the genetics department, and by his continued
promulgation of MYCIN and by serving as coordinator for medical school research
interests in SUMEX.

But this list would eventually embrace a large part of Stanford University,
a network of personal and interdisciplinary connections that constitutes a
seamless web, a treasure for my own experience and recollection, -- but one that
is perforce hard to fairly acknowledge, and even harder to sever myself from.

Fortunately, the communications net offers a way to soften that severance,
and I will seek every opportunity to use it to stay in the closest contact with
the affairs of SUMEX-AIM that the duties of my new situation allow. My continued
association with Stanford and with SUMEX-AIM ought to be a self-exemplifying
demonstration of the capabilities for community-building and for sustaining the
human relationships in scientific effort that have been our highest hopes for
these new, high technologies.

The realism of these expectations has been substantially tested already in
the way that the Executive Committee of SUMEX-AIM, the user community, and Bill
Baker and his colleagues at NIH/BRP, have been able to work together effectively
and constructively in making this enterprise truly a national resource.

I look forward to continuing to be a part of a team like that!

2.1.2   TECHNICAL PROGRESS

The following material covers SUMEX-AIM resource activities over the past
year in greater detail. These sections outline accomplishments 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 4
beginning on page 61.

2.1.2.1   FACILITY HARDWARE

Over the past year, several significant changes have been made to the SUMEX
hardware configuration and associated system software:

1) Core memory was doubled by adding 256K words
2) The file and tape system hardware was upgraded
3) A connection to TELENET is being implemented

The memory and file/tape upgrades have substantially improved system throughput
and efficiency as discussed below. The TELENET connection is being established
for evaluation as a possibly more cost-effective means for meeting community
communication needs (see page 16). The current system hardware configuration is
diagrammed in Figure 1 on page 9.

INTRODUCTION

The SUMEX-AIM facility has been operating at capacity in terms of prime-
time computing throughput and user file space for the past 2 years as documented
in our annual reports (see for example pp 4-8 of the 1976 report). This
condition has constrained the growth of the AIM community and our ability to
bring AI programs nearing operational status in contact with the potential
external user communities while continuing to support on-going program
development efforts. We have taken active steps to try to transfer prime time
loading to evening and night hours including shifting personnel schedules
(particularly for Stanford-based projects), to control the allocation of CPU
resources between various user communities and projects, and to encourage jobs
not requiring intimate user interaction to run during off hours by developing
batch job facilities. Despite these efforts, our prime time loading has remained
very high. Perhaps the most significant effect of the resulting poor response
time is the deterrence of interactions with medical and other professional
collaborators experimenting with available AI programs, whose schedules cannot be
adjusted to meet computer loading patterns (see for example the MYCIN report in
Section 4.2.6 on page 163).

Two years ago, the Executive Committee gave approval for the augmentation
of SUMEX-AIM computing capacity by adding a second CPU. The decision for the CPU
was made as a trade-off between adding memory and/or CPU to maximize capacity
enhancement within the resources available (see the 1976 annual report for a
discussion of these trade-offs). We implemented the dual processor system in the
spring of 1976 and brought it on line in June. The additional capacity was put
to use very quickly as reflected in system usage and loading data summarized in

Figure 6 through Figure 8. 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 substantially increased throughput. The "tolerable" peak load average increased, the number of jobs on the system increased, and the number of delivered CPU hours increased. At the same time (as predicted) the overhead per machine rose dramatically as shown in Figure 9 and Figure 10. The overhead increases came principally in the category of I/O wait (total scheduler time and time waiting for a runnable job to be loaded in core) and in the time processing pager traps. Another 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.

After the dual processor augmentation, SUMEX-AIM computing capacity again became overloaded. This continuing saturation has raised serious discussion about the scope of computing needs of the AIM community and possible justification of additional PDP-10 scale machines to be added to the AIM network. Several specific proposals have been submitted for additional user nodes. We expect additional capacity to be available through the Rutgers resource by the end of this summer and support expansions at other AIM nodes as justified by local and community needs. From the SUMEX viewpoint, we have attempted to do everything feasible and economically justified within current budgets to maximize the use of the existing hardware for productive work. After the dual processor augmentation, the obvious remaining CPU resource to be tapped was to reduce the high dual processor overhead.

A parallel saturation problem existed for a long time in file space commitments. We had queued requests from numerous projects for increases in file space including INTERNIST, Higher Mental Functions, Language Acquisition Modeling, DENDRAL, Chemical Synthesis, MYCIN, and several pilot projects. We did not have additional space to allocate to meet these needs and our DEC RP-10C controller configuration was full (7 drives on-line and one available for backup). We had taken an active role in trying to optimize use of available space by limiting the total space available to projects, limiting the number of versions of experimental files kept on the system, and encouraging the use of tape or Datacomputer archive services for files not needed on-line routinely. We still were unable to adequately provide for the growing needs of existing projects or meet the bare space needs of new projects getting started.

The following plan was presented to the Executive Committee to increase the capacity of the SUMEX facility configuration by 1) adding memory to optimize dual processor utilization and 2) redesigning the file system (including the tapes used to backup and archive user files) to meet increased demands within up-to-date technology. This plan was approved in June 1977 and implemented in September 1977.
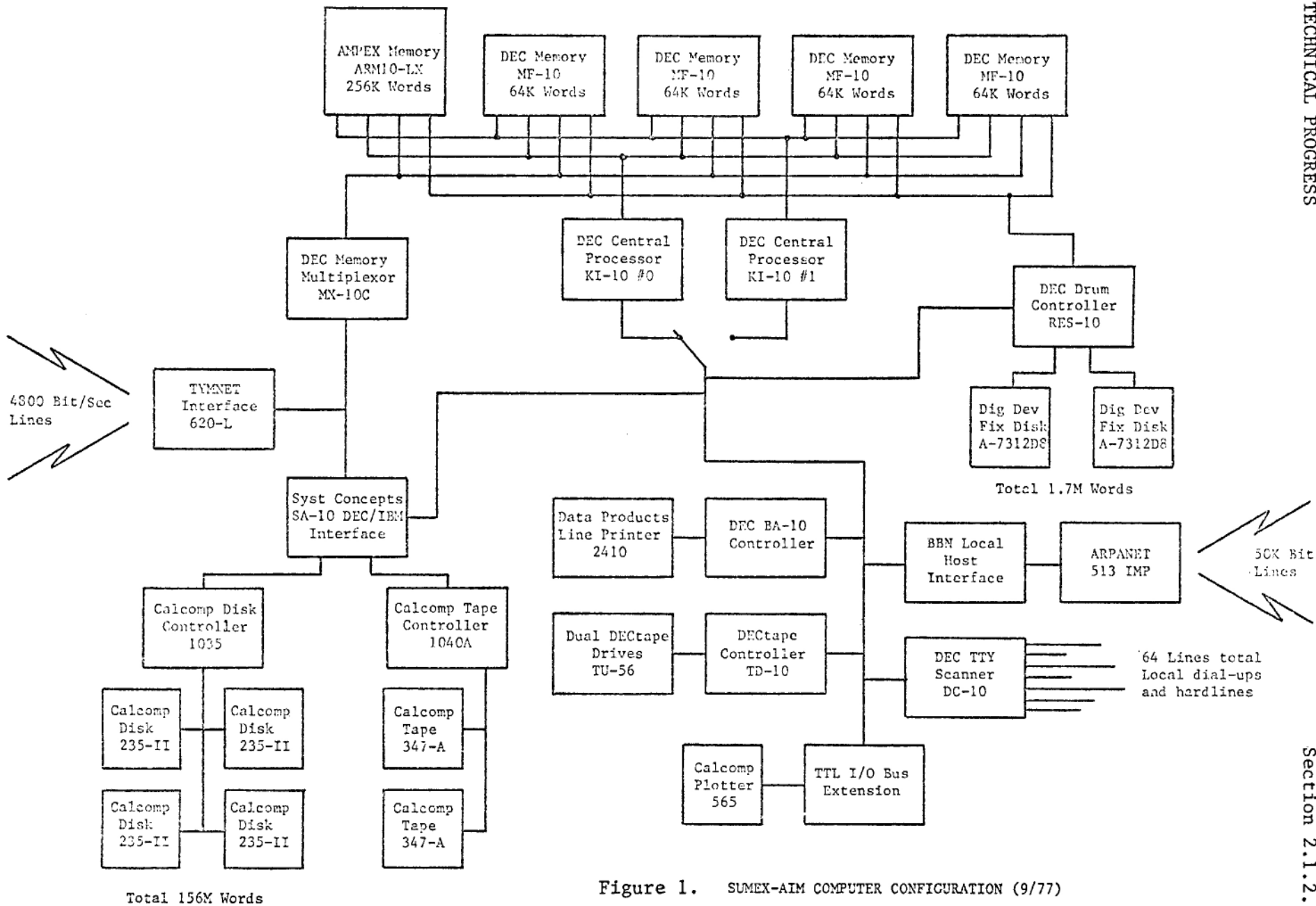
AMPEX Memory
ARM10-LX
256K Words

DEC Memory
MF-10
64K Words

DEC Memory
MF-10
64K Words

DEC Memory
MF-10
64K Words

DEC Memory
MF-10
64K Words

DEC Memory
Multiplexor
MX-10C

DEC Central
Processor
KI-10 #0

DEC Central
Processor
KI-10 #1

DEC Drum
Controller
RES-10

4800 Bit/Sec
Lines

TYMNET
Interface
620-L

Dig Dev
Fix Disk
A-7312DS

Dig Dev
Fix Disk
A-7312DS

Total 1.7M Words

Syst Concepts
SA-10 DEC/IBM
Interface

Data Products
Line Printer
2410

DEC BA-10
Controller

BBN Local
Host
Interface

ARPANET
513 IMP

50K Bit
Lines

Calcomp Disk
Controller
1035

Calcomp Tape
Controller
1040A

Dual DECtape
Drives
TU-56

DECtape
Controller
TD-10

DEC TTY
Scanner
DC-10

64 Lines total
Local dial-ups
and hardlines

Calcomp
Disk
235-II

Calcomp
Disk
235-II

Calcomp
Tape
347-A

Calcomp
Disk
235-II

Calcomp
Disk
235-II

Calcomp
Tape
347-A

Calcomp
Plotter
565

TTL I/O Bus
Extension

Total 156M Words

Figure 1. SUMEX-AIM COMPUTER CONFIGURATION (9/77)

J. Lederberg & E. Feigenbaum

MEMORY AUGMENTATION

There is a close interaction between memory size, CPU capacity, and secondary swapping storage in determining the performance of a demand-paged system like TENEX (see 1976 annual report pp 4-10). Our system as initially designed was quite well balanced in these respects. As the SUMEX-AIM computing load reached capacity, the choices for augmentation dictated either memory or CPU as we had insufficient funding to augment both. We chose CPU as the most user-effective means of providing more capacity at that time. However, as pointed out then, the added CPU power has the effect of increasing the system overhead in order to manage the increased number of jobs using the system within available memory. This shows up in increased pager trap time, interrupt-handling overhead for drum swaps, requirements for additional secondary swap space to accommodate the added jobs, and I/O wait time to fetch a runnable job into core as illustrated in Figure 9 and Figure 10.

Recorded data on dual processor performance show that, during prime time loading, the overhead in I/O wait time, pager trapping, and drum interrupt handling effectively amounted to about 40-50% of the second CPU (actually all of the I/O traffic is on one of the machines whereas other overhead for paging and I/O wait is distributed between them). This lost capacity was recoverable by adding memory. The effect of increasing memory is to allow more jobs in core at once (larger "balance set") and larger working sets to reduce overhead factors. This not only improves efficiency but smooths out user interaction since the larger balance set makes it more likely that pages for a given job will be in core when needed for teletype service. The 256K memory configuration only afforded a balance set of 4-5 jobs, so that with a load average of 7-10 during prime time, about 4-5 jobs had to be completely swapped out at a given time and hence could not get any service. The larger balance set means fewer jobs swapped out for a given load average.

The added memory size also allows more effective use of slower swapping space, particularly with the parallel disk system upgrade to the faster 3330 technology as outlined below. Having more jobs in the balance set makes it more likely that a runnable job is there and reduces the page fault rate so that swapping between memory and the slower store can occur without loss. Time previously wasted as I/O wait to get a disk-swapped job back into memory is reduced.

We considered a number of memory vendors and also looked into a new "slow" AMPEX memory which trades speed (3 usec versus 1 usec) for capacity (1000K words versus 256K words). This memory could be configured either in the form of a random access memory (RAM) or a block transfer ("drum-like") device. In essence the "block transfer" mode would add another layer in the hierarchy of storage intermediate between drum and high speed memory. We felt this type of "drum" memory would not be the most advantageous solution as the system was already burdened waiting for runnable jobs to execute and handling a high overhead of page swapping. The CPU time for the management of page swapping is non-trivial and, based on measurements of swapping activity, the overhead in managing storage on the KI-10 rivals the rotational latency of the drum so the fast swapper would not do that much good. The dominant factor in the overhead is that we had a relatively small executing store for our processing capacity so that under heavy loads the system thrashes trying to service runnable jobs for all the users.

Similarly, configuring such slow memory as RAM for our non-cache KI-10's would slow the processors down by a factor of 3 when executing code from the slow memory. It would be equally costly to rearrange pages between fast and slow memory. Without a special transfer device, the CPU would have to do the transfers limited by the slow memory speed. For a cache system (like the KL-10), this problem may be overcome by the firmware management of the movement of active memory locations between the very fast cache memory and the slower memory (4-word parallel transfers).

Thus we felt the most effective remedy was more high speed memory. We chose AMPEX memory from among the vendors reviewed as the best trade-off between performance, price, packaging, and maintainability. The additional 256K of memory was brought on-line in September 1977. From the data shown in Figure 9 and Figure 10, it is clear that the predicted reduction in system overhead was immediately achieved. The following table shows measurements of average instruction times comparing our AMPEX and DEC MF-10 memories. Also included for comparison are data for a Systems Concepts memory installed at the IMSSS KI-10 facility at Stanford:

	AMPEX	DEC MF-10	SC MF-10
MOVEI	1.20	1.21	0.97
MOVE	1.54	1.64	1.24

These data give the time in microseconds to execute the instructions shown based on the DEC timing diagnostic and normalizing the times to a "standard" 15 foot memory cable length. The MOVEI instruction shows the relationship between the basic memory access times and the MOVE instruction illustrates the effects of KI-10 "look ahead" with overall memory cycle time. Currently our AMPEX memory is timed to have essentially the same access time as the MF-10's but it is actually capable of somewhat faster operation. We are planning to attempt to reconfigure the memories this summer to take better advantage of the AMPEX speed. This may recover about 80 nanoseconds per access. This will still not bridge the timing difference between the AMPEX and the Systems Concepts memories. Systems Concepts offers a technically advantageous memory in terms of speed. Our choice of vendors was based on our own evaluation of issues like resale potential, maintainability, and management responsiveness, taking into account our experience with Systems Concepts in purchasing their disk channel interface (see below).

It should also be mentioned that the installation of the additional 256K of memory required modifications to the MX-10 memory multiplexor to accommodate 22-bit addresses and to the TYMBASE to be able to operate on a KI-10 style memory bus supporting more than 256K of memory.

DISK/TAPE RECONFIGURATION

Disk technology has changed rapidly in recent years. At the time we bought the initial SUMEX configuration, taking into account the discount DEC gave on the system purchase and maintainability, the DEC RP-03 system we bought was the best choice. Since then double-density 3330 technology has become well established (prices for IBM-compatible equipment were cut almost in half in 1976 alone!) and even higher densities are coming along. Given the relatively low incremental

cost (for used equipment), we added RP-03 drives until filling the capacity of
the controller. But with the added demands of community projects, a better long
term solution necessitated upgrading from the RP-03 technology. Newer devices
offer more economical future growth, and faster transfer rates thereby further
decreasing system overhead.

Our tape system was in an even more advanced part of the age curve. We
have not emphasized individual user tape services at all but tapes are critical
to system operation for file system backup and user file archiving. We minimized
the initial investment in tape drives to the advantage of other parts of the
system. To accommodate the larger file system and to improve system operations
and efficiency, the upgrade in file system also required a parallel upgrade in
tape service. An additional advantage to upgrading the tape system was to move
the I/O interface from the I/O bus to a direct memory interface thereby reducing
system interrupt loading during prime time tape/file system operations.

The most attractive approach to file/tape system upgrade was to adapt a DEC
memory port to look like an IBM selector or block multiplexer channel and then to
take advantage of the substantial price competition in the IBM-compatible
peripheral market. The capacity of a double-density 3330 disk drive is equal to
4 RP-03's. Thus bringing 3 new drives on line almost doubles the on-line
capacity. After investigating alternative vendors, we selected a system using a
System Concepts SA-10 channel adaptor, Calcomp 235-II disks, and Calcomp 347A
tapes. This system was installed and brought on line at the same time as the
memory augmentation in September 1977.

•This system has substantially alleviated file capacity pressure and made
possible much smoother backup operations. With the faster tape speed, we no
longer take the system down for pack copies Sunday morning but rather do a full
file system dump to tape. Similarly during the week we do incremental dumps back
to the previous full dump each day to give quite good backup coverage. We have
experienced no major technical problems with the new file/tape system; more
details about impact on system software is given in a later section.

Unfortunately, we have experienced many frustrations dealing with Systems
Concepts management in contrast to the high technical quality of their hardware.
There remain several parts of the SA-10 adaptor that have not been delivered
including full documentation, maintenance training, cabling to replace that which
we borrowed for installation, and the device indicator panel. This experience
led us not to consider Systems Concepts for memory. Few such memories have been
delivered and it is not clear that we could depend on future maintainability. On
the other hand, support for the SA-10 is secure in that many are in the field
with excellent service records and alternative sources exist for SA-10
maintenance through Calcomp, DEC, or TYMSHARE.

2.1.2.2   SYSTEM SOFTWARE

MEMORY EXPANSION AND FILE/TAPE UPGRADES

The addition of 256K of memory and the upgrades of our file/tape system
necessitated a number of changes in the monitor. TENEX had not fully anticipated
memory addresses longer than 18 bits and so those places where half-word

addresses were assumed had to be fixed. The RP-10C disk service and TM-10 tape service code had to be replaced by code that produces the appropriate IBM channel commands. We imported the "standard" BBN SA-10 disk/tape service and incorporated it in our dual processor system. Despite the substantial amount of work required to incorporate this code into our system, the new hardware and monitor came up smoothly on 9/1/77.

We have encountered a number of bugs during the year, particularly in the disk service. The most troublesome one resulted in a deadlock between command retry attempts from the CPU and a busy controller state. The 3330 recovery procedure implemented in the BBN code appeared to track exactly that used in IBM's most recent VS releases. Nevertheless infrequently during internal controller error correction attempts, we found the system hung in a loop with the controller, when trying to restart commands queued at the time of the error. It appears the problem may be in the Calcomp controller microcode but we have not been able to get enough information from Calcomp to confirm that. Meanwhile we have constructed a software work-around to detect the loop when it occurs and to reset the disk channel before proceeding.

The new hardware had other ramifications for system software as well. DEC's diagnostic system is designed to run off of their disk or magnetic tape systems. This capability was lost as a result of the change from DEC hardware so that diagnostics had to be loaded from slow DECtape units. We have invested considerable effort in bringing system diagnostic facilities back up to a workable level; borrowing programs others had written and implementing new ones where needed. We have implemented a stand-alone facility to load SAV files from the TENEX file system, incorporating full TENEX name recognition features. This means that programs can be manipulated and kept on-line in the time-sharing file system and then loaded as needed when the machine is down or in stand-alone mode. This also provides an easier way to reload the monitor. We have written a fast disk pack copy routine for the 3330 packs and have improved the SA-10 diagnostic package to check out 512K of memory and to ensure safer testing of disk drives in the presence of live file system packs.

Also with 1600 BPI tapes available changes were necessary to the tape service and TOPS-10 compatibility package to accommodate DEC's extended magnetic tape UUO's as well as to be able to fully use the byte packing facilities of the SA-10 and IBM drives.

We continue to work to improve the efficiency of the system and its effectiveness in allocating valuable resources. We have implemented a high priority hardware clock to sample monitor and user mode program counter locations to find places of abnormally high activity and perhaps inefficiency. This has pointed out several "hot spots" in routines where it was obvious on other grounds that a substantial amount of time is spent (e.g., drum service, KI page handling and teletype service) but there are no clear solutions to these problems with the current hardware.

SYSTEM LOADING CONTROLS

We previously 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 editing, 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 consumed during a given allocation control 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 the system. It allocates cycles preferentially, trading off a priori allocations with actual demand but does not waste cycles.

This scheduling scheme does not deal with the problems of system overloading during peak periods. At such times (mid-morning and mid-afternoon especially), one observes what has been termed "the tyranny of time-sharing". System efficiency and user response time degrade because the system is trying to serve more jobs than it has reasonable resources for. Users sit at their terminals waiting for the cycles they need to work effectively but there are not enough to go around. Ideally the system should have a response time keyed to a typical human interactive response interval. This implies a limit on the number of active job slots that the system can accommodate simultaneously in order to approximate this ideal. In some systems a "pie-slice" scheduler is used wherein a group of users is allocated some percentage of the machine and if that group consumes more than that amount during the cycle interval, its jobs are not scheduled until other groups catch up. Meantime, those users sit at their terminals and receive VERY slow response, not knowing when things will let up so they can effectively compute again. This type of approach does keep the system from trying to run too many jobs at once but it does not solve the problem of EFFECTIVELY MANAGING USERS' TIME.

We have attempted to control system overloading in a somewhat different way to better manage user time and to allow us to better apportion system capacity between communities and projects during heavy load. Each project gets its pro rata share of the active job slots the system can accommodate. Rather than allow many users to ineffectively vie for each project's slots (as in the pie-slice system), we ask selected users within each group to restrict their use for periods of 1/2 hour so that those remaining can work effectively within the project aliquot. Allocation of active job slots is made on the basis of relative community and project percentage allocations (assigned by the AIM Executive committee). Within each project slots are allocated either on a round-robin basis or taking into account optional project priorities among users. Under overload conditions, active jobs outside of the available slots are asked to slow down, thereby holding the load within tolerable limits. If such jobs do not voluntarily cooperate, they may be forced to comply.

An overload condition is defined to be one in which the overall load average exceeds a threshold (currently 7.5), significantly more jobs are runnable than there is core for, or excessive page faulting is occurring. Outside of periods of overload, the previous "soft" percentage scheduling scheme is applied. Also thresholds for overload conditions may be dynamically adjusted to assure good system response during a demo.

This system has been in operation for approximately one month during which time we have experimented with various threshold adjustments and observed its effects on user behavior. During this early period, we have not placed any controls on system use by the AIM community projects since they have historically been below their quota for system use. It is still early to tell quantitatively what its effect will be; system usage fluctuations are such that we will have to observe operations for several months before drawing conclusions. However, qualitatively it seems to be holding system loading within tolerable bounds and allocating capacity as apportioned between the various communities and projects.

OTHER ENHANCEMENTS

Other areas of system software development include the EXECutive program, the BSYS program for file archiving and retrieving, the printer spoolers, the CHECKDSK program for verifying file system integrity, and numerous smaller utility extensions and bug fixes. We have continued to improve the EXEC in such areas as the DIRECTORY command (to display last file reader and temporary files), the MAP command (to handle long files), a new INITIALIZE command (to restart the EXEC after errors), smoothing out multi-directory search paths using features of the new GTJFN, adding wild card and question mark facilities to the file retrieval INTERROGATE command, making the command for changing file protection more mnemonic, and restoring terminal modes correctly after a forced detach (e.g., with a dropped line or network disconnect).

We have added a facility to the BSYS program to automate the restoration of requested files from tape, avoiding the earlier error-prone and time-consuming typing of individual restore commands. We have completely rewritten the line printer spoolers to more efficiently and uniformly handle the local and remote printers, to add facilities for "unlisting" a file listed by mistake, and improving the marking of listing boundaries to ease operator separation of listings for various users.

We have imported a new version of CHECKDSK initially written at BBN and have incorporated local facilities for more extensive file system integrity checking. This version presorts file index block addresses and scans for errors using more sequential disk I/O. Several forks are started, one for each drive. These keep the disk channel as busy as possible while performing the check computations. This improvement has reduced the time to scan the file system from 20 to 6.5 minutes.

2.1.2.3   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. These include improved
inter-user communications, more effective software sharing, uniform user access
to multiple machines and special purpose resources, convenient file transfers,
more effective backup, and co-processing between remote machines. Until now, 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. Other commercial network systems (notably TELENET)
have come into existence and are growing in coverage and services.

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 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. Limited responsiveness for character-oriented
TENEX interactions continues to be a problem for network users.

TYMNET:

Networks such as TYMNET are a complex interconnection of nodes and lines
spanning the country (see Figure 2 on page 18). 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 seen
little improvement in TYMNET service over the past year although the cost of
service has risen sharply. We purchase TYMNET services through a volume contract
the National Library of Medicine has with TYMNET. The cost has gone from
approximately $2.90 to $6.09 per connect hour. Because of this increase, we are
investigating alternative sources of network service; in particular TELENET.

We have had a number of technical problems with the TYMNET this past year.
Internally they changed some of the protocol involved in the TYMBASE connection
we use. They neglected to tell us about these changes though and the problems
that resulted were very hard to track down. From the user viewpoint, connections
were dropped frequently. From the system viewpoint, we could not tell if the

problems were subtle results of the recent memory and file system hardware change. That change required a modification in the TYMBASE to accommodate the new memory bus conventions. This took months to isolate but the TYMNET interface is finally running reliably again after much user frustration.

ARPANET:

Current ARPANET geographical and logical maps are shown in Figure 3 and Figure 4 on page 19. Consistent with agreements with ARPA and the Defense Communication Agency, we are enforcing a policy that restricts the use of ARPANET to users who have affiliations with ARPA-supported contractors and system/software interchange with cooperating TENEX sites. We have maintained good 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.

TELENET

We recognize the importance of effective communication facilities for SUMEX-AIM users and are continuously looking for ways to improve our existing facilities. A year ago we did some preliminary investigations of TELENET facilities that have been rapidly expanding this past year (see Figure 5 on page 21). 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 features in terms of a symmetry analogous to that of the ARPANET for terminal traffic and file transfers and being a commercial network, it does not have the access restrictions of the ARPANET. As indicated above, the cost of using the TYMNET has increased this past year so that TELENET rates appear to be substantially lower for supporting community communication services. The National Library of Medicine has a contract with TELENET which includes significant cost advantages through combining our use with NLM use to achieve a high volume discount. As a result of discussions with the AIM Executive Committee and BRP, we are in the process of implementing an experimental connection to TELENET with the view of moving SUMEX users to that service if technically effective.
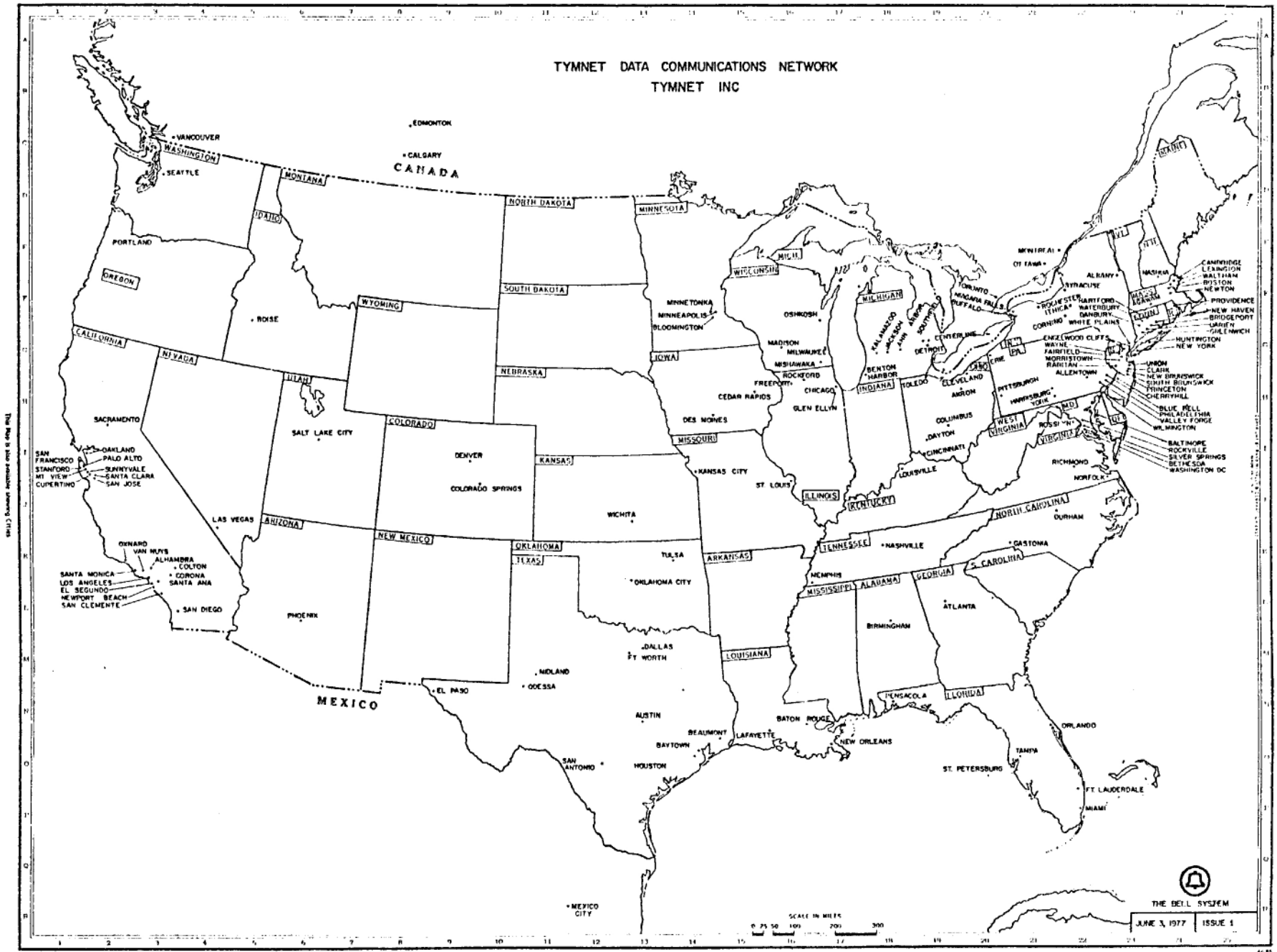
TYMNET DATA COMMUNICATIONS NETWORK
TYMNET INC

THE BELL SYSTEM

JUNE 3, 1977 | ISSUE 1

Figure 2. TYMNET Network Map

ARPANET GEOGRAPHIC MAP, MARCH 1978

∿∿ 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
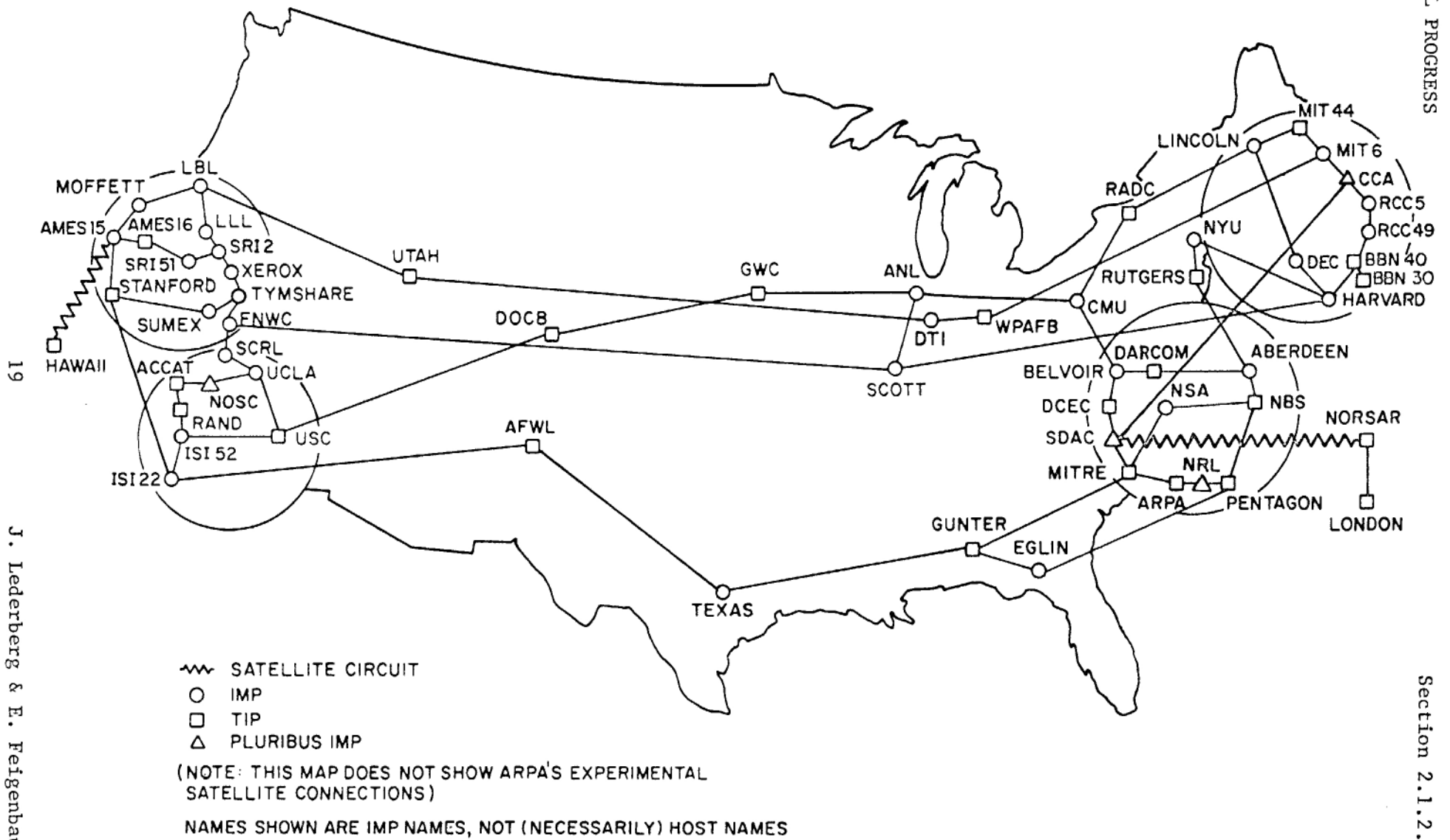
Figure 3.   ARPANET Geographical Network Map

ARPANET LOGICAL MAP, MARCH 1978

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

HOST COMPUTER CONFIGURATION SUPPLIED BY THE NETWORK
INFORMATION CENTER

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

○ IMP
□ TIP
△ PLURIBUS IMP
∿ SATELLITE CIRCUIT
▽ VERY DISTANT HOST
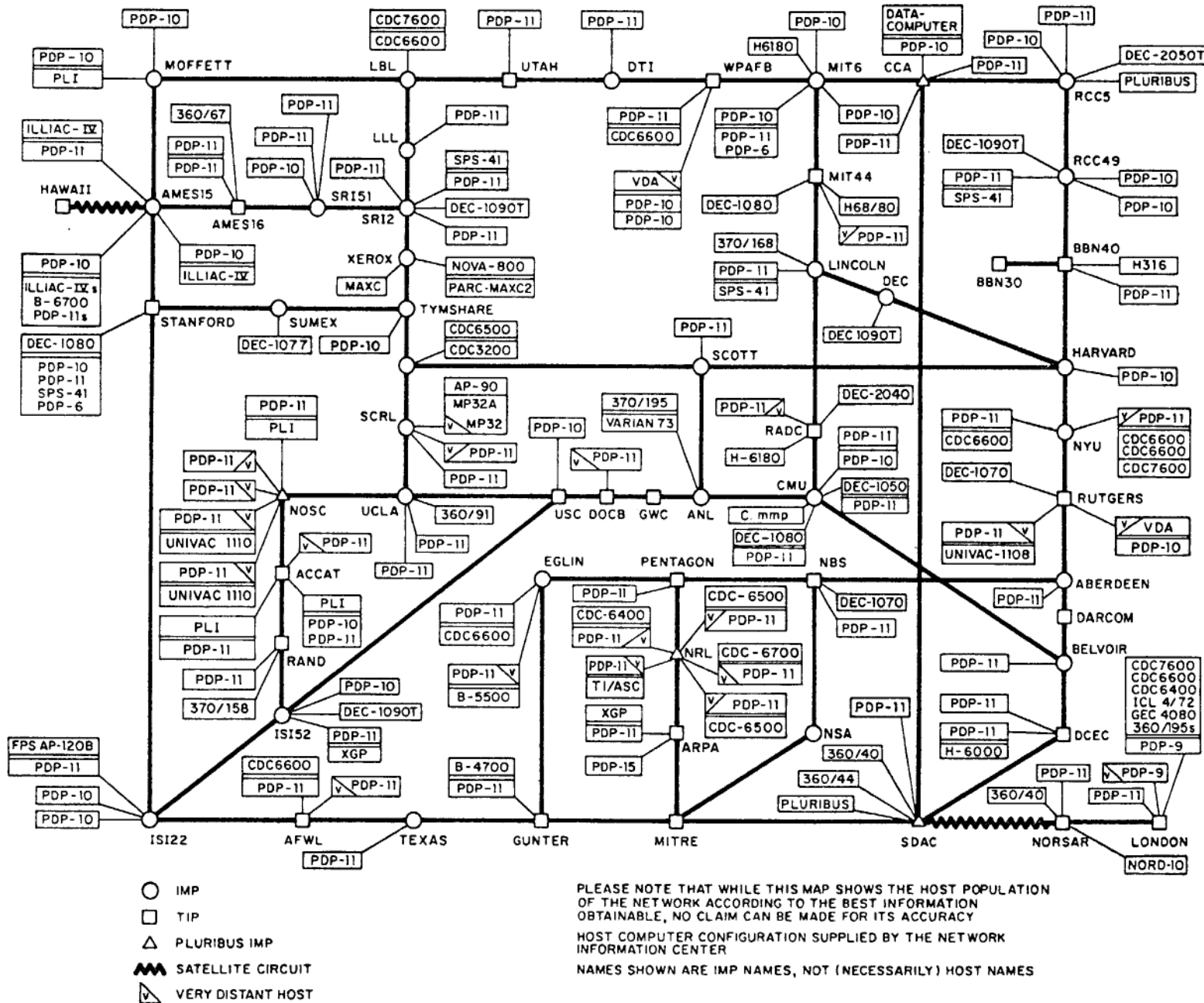
Figure 4.   ARPANET Logical Network Map

TELENET GEOGRAPHIC MAP

MID 1977

TO
EUROPE

TO
PUERTO RICO

21

J. Lederberg & E. Feigenbaum

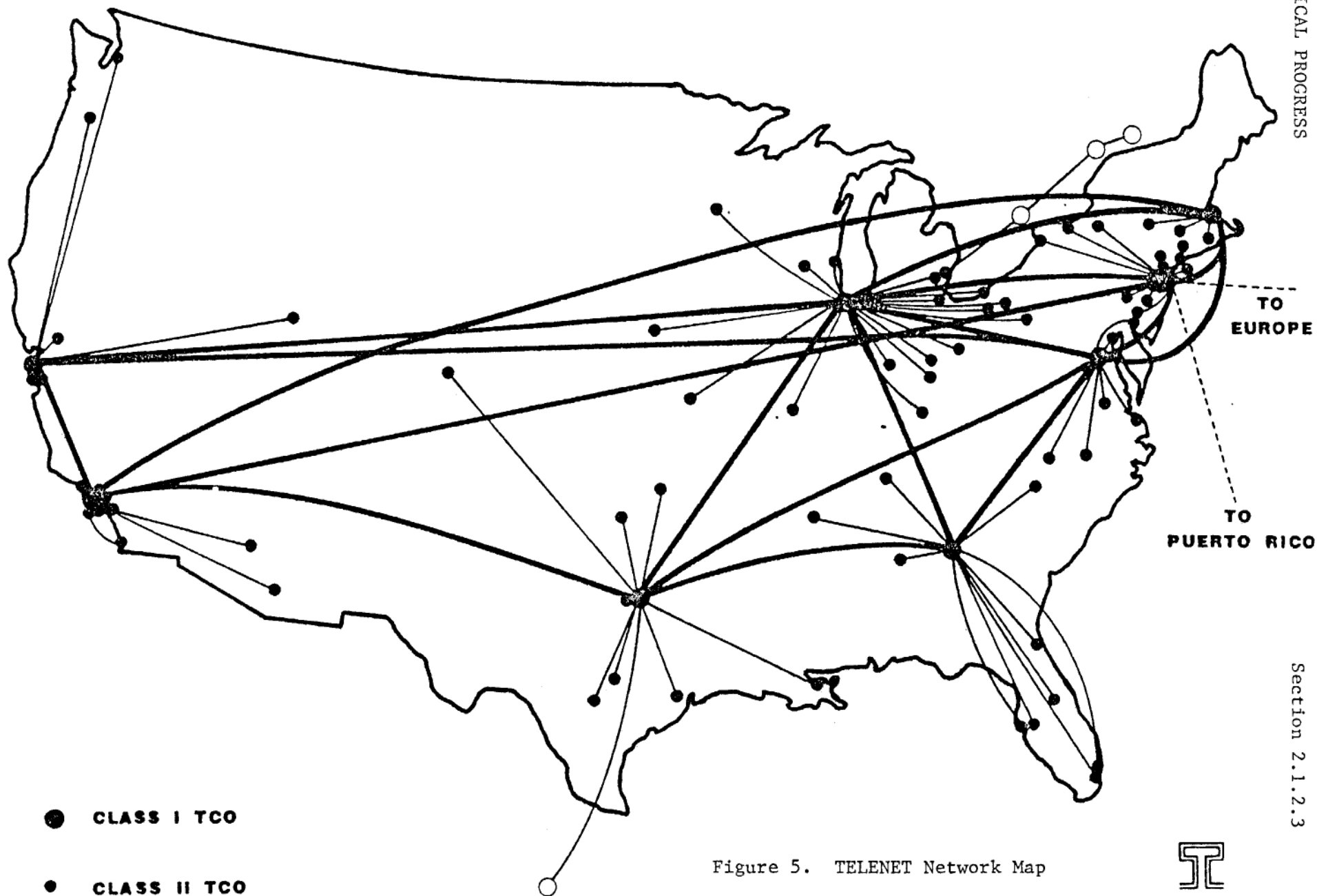● **CLASS I TCO**

● **CLASS II TCO**

Figure 5.   TELENET Network Map

2.1.2.4   SYSTEM RELIABILITY AND BACKUP

System reliability has been very good after the installation of the new memory and file/tape hardware. There have been a number of problems as detailed earlier with the disk system and TYMNET that have caused more crashes and dropped lines than normal. Also in the process of experimenting with speeding up our memory configuration, we have caused some unreliability. The table below shows monthly downtimes for the past year.

	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR
			1977						1978			
CRASHES												
Hardware	12	27	8	18	6	6	14	6	6	5	2	4
Software	0	0	4	4	2	1	3	4	1	0	6	3
Environmental	1	2	2	2	2	0	0	1	1	1	1	4
Unknown Cause	0	2	3	2	3	5	6	5	4	7	3	2
DOWNTIME (Hrs)												
Unscheduled	30	70	34	47	19	19	48	20	17	25	13	15
Scheduled	41	73	25	83	61	44	31	30	23	15	28	31

TABLE 1.   System Reliability by Month

In May and June we experienced a substantially higher number of hardware crashes which we feel resulted from the system being overheated during an air conditioning failure in mid-May. Ultimately an intermittently shorting backplane wire was found as well as several intermittent arithmetic unit failures. During late July and August we worked on system hardware and software changes preparatory to the installation of the additional 256K of memory and the new file/tape system hardware. This increased system downtime and unreliability as well. Infant mortality problems with the new hardware (AMPEX memory especially), caused a number of crashes in August, September, and October. The high number of "Unknown" crashes until recently are the result of a number of factors which were hard to separate or caused the system to fail in ways that we couldn't reconstruct what happened. We feel these resulted from the TYMNET protocol problem mentioned earlier, a race condition between the AMPEX and DEC memories, disk controller problems, and software bugs. Between September and late October we worked on organizing and enhancing system diagnostic capabilities to support the new hardware. This required increased downtime.

2.1.2.5   SOFTWARE EXPORT

Over the past year we have continued to investigate alternatives for software export. The following reports on several of these areas, including 1) the availability of small PDP10-like machines, 2) progress in developing the MAINSAIL language, and 3) an investigation of possibilities for writing a MYCIN-like program using an algorithmic language.

SMALL PDP10-LIKE MACHINES

Early this calendar year DEC announced a new small machine designated the 2020. This machine approximates the "small PDP-10" machine we had discussed in last year's report. It allows up to 512K of memory, 2 RP-06 disk drive, 2 tape drives, a printer, and runs DEC's TOPS-20 operating system. Prices range from $150K for a 128K word machine to $375K for a 512K word machine with two disk drive, tapes, and a printer. The unloaded performance of this machine appears to be in the range of a KA-10 but only preliminary benchmarks have been run. Lynch at SRI has run a simple LISP test program on a range of machines. His test creates a large list, randomizes it, and then sorts it. Using a KA-10 with 512K of memory as a reference, the performance of various PDP-10 systems is shown in the following table.

Function	KA-10 256K	KA-10 512K	2020 512K	KI-10 512K	KL-1090T 1024K
Build List	0.94	1.00	1.20	1.79	5.86
Rearrange List	0.92	1.00	1.13	1.64	5.20
Sort List	0.79	1.00	0.89	1.65	4.56

This test indicates that the 2020 performs at about the same level as the KA-10 for a single user. These data do not give a complete picture of performance under increasing load, however, and do not fully reflect the intrinsically slow arithmetic performance of the 2020.

We have attempted to run benchmarks of the CONGEN and MYCIN programs to compare these machines. We ran squarely into a compatibility problem however. We prepared the two benchmarks, ran them on our KI-TENEX system, checked to see that they would run on TOPS-20 using SRI's 1090T system, and asked DEC to run them on the 2020. The benchmarks failed to execute because of some system call changes DEC had just made to the newest release of TOPS-20 running on the 2020. We are just now getting access to a machine running that version of TOPS-20 and hope to fix the incompatibility to complete the benchmarks. This experience reinforces our belief that increasing incompatibilities will show up between TENEX and TOPS-20 that will make software transfer difficult.

We have had a number of contacts from outside users interested particularly in the chemistry AI programs. Such a machine would represent a good solution for such groups to gain access to the programs, maintain the necessary security for proprietary data storage, and stay abreast of new developments. This type of approach is also attractive for providing needed capacity expansion in small increments within the AIM community (e.g., for more extensive testing of the MYCIN or INTERNIST programs) while maintaining general software compatibility. Remote location of such machines within the community may also offer significant advantages for human interfaces since terminal handling can be done locally thereby supporting higher speed lines and improved echo interactions for recognition, etc.

MAINSAIL

During this past year we have concentrated on six areas:

1) Implementations
2) Runtime design
3) Language design
4) Compiler design
5) Documentation
6) Emulation research

We have not yet extensively distributed MAINSAIL since it is still undergoing development based on our experiences with it locally. We have continued to receive many inquiries concerning the progress of our work, with several projects considering using MAINSAIL when it becomes available.

At present our major concern is MAINSAIL's efficiency in a small address space; in particular, the compiler cannot yet run on a PDP-11. Though it appears that computer technology is moving towards large address spaces, existing machines with 32K-word address spaces will persist for many years, and many people have indicated an interest in using MAINSAIL on such machines. The difficulty is that MAINSAIL provides features which are not easily supported when memory is scarce. Over the past year we have gained a better understanding of MAINSAIL's resource requirements, and have taken steps to reduce its implicit use of memory.

Implementations

We have developed five implementations for two computers: TOPS-10 and TENEX for the PDP-10; and RT-11, RSX-11M and UNIX for the PDP-11. The last two were developed during the past year. The others have received varying amounts of bug fixes and updates. The TENEX version has been in use for about two years, and the RT-11 version for about a year. The TOPS-10 version has been used to a lesser extent for about a year. Programs have been run on RSX-11M and UNIX, but these implementations are not complete.

No implementation is in general use; in some cases they have primarily served to insure that the runtime design is sufficiently flexible. Each new implementation has revealed deficiencies in the design which have since been corrected. We will need to implement MAINSAIL on some non-DEC machines before we can get an unbiased assessment of the difficulty of creating new implementations.

Runtime Design

A new runtime system is now under implementation. It is oriented towards execution efficiency and less memory utilization since these are the problems with the current PDP-11 implementations.

A major savings has been made with regard to string constants. In the previous implementation, the text of string constants was copied into string space where it remained throughout execution. Also, the string-constant descriptors were allocated in the data sections, which remained in memory as well. In the new implementation, the string-constant text remains in the control

section, so that it is swapped out of memory along with the control section.
String-constant descriptors are created each time a string constant is used.
This usually requires that the text for a string constant be copied into string
space upon each use. The overall result is that string constants do not tie up
memory as in the previous implementation, but more time may be spent repeatedly
copying string constant text into string space. This could also lead to more
string "garbage" collections.

The implementation of modules in terms of control sections, data sections,
and descriptor sections has been altered to save memory. Procedure call, entry,
exit and return have been redesigned to save code and time. The amount of code
executed for i/o has been decreased. A new approach to the use of "anonymous"
modules has been implemented, and the manner in which modules obtain linkage to
one another has changed.

The previous implementation required that every module reside in a separate
file which is opened and closed during execution in order to access the module's
code. The new implementation provides "runtime libraries" which are files
containing any number of modules. Each runtime library remains open throughout
execution. There will be a standard runtime library containing the system
modules, and another containing the compiler modules. The programmer may also
contribute runtime libraries. Single-module files remain as before.

The size of the "kernel" module (which is always resident) has been
decreased. There is more reliance on incremental initialization of arrays,
string space, string constants, class descriptors, module pointers and module
descriptors. This allows some code to be moved out of the kernel into separate
modules.

The modules which make up the runtime system have been reorganized to
decrease the number of costly intermodule calls. Each module is relatively more
self-contained. In the previous implementation, many calls to system procedures
resulted in a chain of intermodule references which resulted in thrashing on the
PDP-11. In most cases a call to a system procedure now requires at most a single
system module.

A preliminary version of a debugging module has been written and utilized
to some extent.

Language Design

There have been some changes to the language, primarily to support the new
runtime implementation.

OWN arrays are no longer handled any differently than other OWN variables.
An OWN array's allocation is now under programmer control. An OWN array's
declaration may no longer include initialization values. Instead, an INIT
statement is provided which can initialize any array with constant values.

To reduce the number of intermodule calls, and thus the amount of potential
swapping and the extra code executed for the calls, the concept of "compiletime
libraries" has been introduced. A compiletime library is a file containing
procedure bodies that are to be "compiled into" a number of different modules