M.I.T. Laboratory for Computer Science June 11, 1980

Computer Systems Research Division Request for Comments No. 189

DRAFT OF ANNUAL REPORT, July, 1979 - June, 1980

Cover note by J. H. Saltzer

Attached is a rough draft of the Annual Progress Report of the activities of the Computer Systems Research Division, assembled by concatenating with only minor editing submissions from quite a number of group members. Please look it over and offer comments on

- overall organization
- omissions of significant activities
- details that are wrong
- anything else.

Check also the lists of papers, talks, committee memberships, etc., that appear at the end for mistakes and omissions.

The final version of this report will be included in the L.C.S. Annual Report, and we will also make copies for handout to visitors until the L.C.S. Annual Report is available. It is usually the case that the report gets distributed widely; many people follow our activities almost exclusively by this mechanism. Thus there is a significant payoff to making it good.

Incidentally, the final version will probably be ready by the end of June. It would be better to wait for that version to appear rather than giving copies of this one to visitors.

COMPUTER SYSTEMS RESEARCH

Academic Staff

J.H. Saltzer, Group Leader
D.D. Clark
F.J. Corbató
I. Greif
D.P. Reed
L. Svobodova

Research Staff

J.N. Chiappa
E.A. Martin

Undergraduate Students

L. Allen
R. Baldwin
G. Balikrishnan
A. Chiang
D. Daniels
K. Khalsa
J. Lucassen
C. Ludwig
J. Marrgraff
A. Mondori
P. O'Donnell
H. Peterson
M. Plotnick
W. Rubin
G. Simpson
M. Smith
G. Stathis
D. Theriault
J. Thomas
S. Toner
R. Vieraitis
E. Wylen

Graduate Students

W. Ames
G. Arens
E. Ciccarelli
G. Cooper
W. Gramlich
S. Kent
V. Ketelboeter
L. Lopez
A. Luniewski
A. Mendelsohn
B. Myers
R. Schiffenbauer
C. Seaquist
V. Singh
K. Sollins
P.T. Tung

Support Staff

R. Bisbee
D. Fagin
J. Jones
E. Poithier
M. Webber

Visitors

N. Natarajan
M. Sinha

COMPUTER SYSTEMS RESEARCH-DRAFT

1. LOCAL NETWORK TECHNOLOGY RESEARCH

Local network technology in the United States is dominated by variations on the passive broadcast cable pioneered by the Xerox PARC ETHERNET. An alternative technology, the ring of active repeaters, has received less attention, even though it offers a number of attractive properties, including simpler analog engineering, ability to cover a larger geographic area, ability to use fibre optics, and ability to scale up to very high speeds. The laboratory has a modest project underway to explore this alternative in more depth, and to learn more about the properties of the ring network in the field.

1.1. Prototype Ring Experience

A prototype ring network, running at a data transmission rate of 1 Mbit/sec has now been in operation in the laboratory for 18 months, currently connecting eight PDP-11, LSI-11, and VAX computers including a bridge to the other local networks. This basic ring structure has proven to be quite effective in day-to-day use, although the need for automatic reconfiguration when nodes are taken down has been clearly demonstrated. (Reconfiguration in the prototype ring is done manually from a central location.) A second copy of the prototype ring was installed at UCLA in November, 1979, and has been operating there quite effectively, also. (So far, neither of these installations has stressed the ring capabilities enough to provide convincing demonstration of the concept, though.)

As part of the evaluation of the prototype ring, an undergraduate thesis was completed that involved implementing a network performance monitor and collecting an initial set of data. Statistics and operational characteristics quite similar to those reported for the Xerox PARC Ethernet were observed.

1.2. Version 2 Ring

In conjunction with a subcontractor, Proteon Associates, Inc., the prototype ring network was re-engineered around a simpler design and for a transmission rate of 8 Mbit/sec to produce what is called the Version 2 ring. The hardware for the version 2 ring is designed to fit into a general system for a local area network that can cover a site such as the M.I.T. campus.

The key to this system is definition of a high speed byte-parallel local network

interface that permits, on one side, implementation of any of several local network technologies, and on the other, implementation of buffered channel or bus attachments for any of several computers.

Thus the version 2 ring controller comprises a modem, clock circuits, token and ring format management, all on a 5 by 8 inch card containing about 30 TTL integrated circuit packages; it attaches by connector to the standard interface. One lesson learned from checkout of the prototype ring was the value of builtin checkout features; the version 2 ring controller includes a 10-bit shift register that can be connected from the transmitter to the receiver in place of the rest of the ring, allowing local checkout of almost all features of the controller.

Clock coordination is probably the single hardest problem to accomplish in a ring when the goal is to avoid dependence on a central or special station. Agreement on the exact frequency of data transmission must somehow be reached collectively. (In both the Cambridge University and Toshiba high speed ring networks a central station sets the clock rate.) Two schemes have been investigated, with special interest in their stability at high data rates and with large numbers (say 200) of nodes. The initial implementation uses a frequency-adjusting phase-locked loop in each node, comparing the observed received data rate with the local clock and fine-tuning the local (crystal) oscillator to match. A string of repeaters thus would all synchronize their clocks to the frequency of the first node in the string; a closed ring will home in to a communally-agreed-upon average frequency, with the possibility of oscillation around that frequency that can apparently be damped by appropriate choice of filter values in the individual phase-locked loops. Two simple, first-order mathematical analyses predict that stability is easily accomplished; field experience will be required to learn how closely these first-order models reflect the actual operating environment.

A second clock coordination scheme, extrapolated from the scheme used successfully in the prototype ring, is also being investigated. In this second approach, the local clock of each node runs at some modest multiple, say 6X, of the nominal data transmission rate, but its frequency is fixed. Received data is examined and its clock rate and phase extracted and compared with that of the transmitter side of the node. If the transmitter phase drifts more than, say, 1/6 of a bit time ahead or behind the received phase, the transmitter sends one bit that is either 1/6 of a bit time shorter or longer than usual, so as to catch up. This approach has the virtue that it is largely digital in nature, can correct much larger frequency errors, and does not require continuous transmission. However, for stability it requires that between messages there should be gaps with no transmitted data, which in turn requires the receiver of a node be able to decode incoming data starting with the first transition of a sequence of bits; at high frequencies and in the presence of noise, this rapid startup is relatively hard to accomplish.

As mentioned above, the frequency-adjusting, closed loop design is being used in the initial implementation. As a second, parallel effort, the phase-adjusting scheme is being tested for its potential applicability.

The frequency-adjusting modem, data transmission over 800 feet of twinax cable, and the ring controller have all been demonstrated individually and in a 2-node ring and their successful integration is expected to be imminent.

1.3. Other Local Area Network Components

As mentioned, the version 2 ring is designed as part of a general, modular system for a local area network. Several other components of this system have been imagined, designed, or implemented. On the host computer side of the byte-parallel net interface, a full-duplex, buffered, direct memory access module for the PDP-11 UNIBUS was specified, designed, implemented, and checked out. Two copies of this 100-chip card have now been built. Similarly, a buffer module for the S-100 bus has been specified, and design begun, and buffer module implementations are planned for the nu-bus and the Q-bus; recently a proposal to implement a buffer module for an IBM 370 channel was discussed.

On the other side of the byte-parallel net interface, design has just begun on a "long-distance bridge" module, which would allow interconnection of local nets in different buildings. The initial version of this bridge will probably use the same basic ring control strategy as the version 2 ring with minor specialization to the case of two nodes and long cables; options such as fibre optic technology are also being examined.

Other possible network technologies that could easily be attached as part of this same system include a packet radio network for communication with computers located in private homes, an Ethernet based on the recently announced standard agreed upon by Xerox, Digital Equipment Corporation, and Intel Corporation, and an X.25 interface to TELENET or TYMNET. Each of these, in turn, could be directly attached to any host for which a host-specific buffer module had been implemented.

2. PROTOCOL DESIGN AND NETWORK INTERCONNECTION

Currently, there are five different network technologies deployed or under development in the laboratory, and four different protocol families in use, with more on the horizon. This excessive wealth of material raises problems of substantial theoretical interest, which must be immediately solved if we are to provide any sort of stable service to the laboratory community.

As a practical matter, the proliferation of network hardware is less disruptive than the proliferation of protocols. Our assumption has been that experimentation with network hardware technology is healthy and appropriate, but that protocol standardization is important if the various machines in our laboratory are to be able to communicate. Thus, we have been attempting to standardize the laboratory on the protocol family developed by the ARPA internet working group, variously called internet or transmission control protocol (TCP). Implementations of these protocols have either been implemented or imported for the Multics system, UNIX, Tops 20, and the Alto. The Alto implementation is coded but not debugged, the other implementations are operational, at least for friendly users. The function of this protocol is to permit traditional services such as remote login, file transfer, and mail to operate in the local environment. Our group has also specified an extremely simple file transfer protocol, as a interim measure until the TCP based file transfer protocol is generally available. This protocol, called trivial file transfer protocol (TFTP), has been implemented for Multics, UNIX, Tops 20, the Alto, and as a stand alone program suitable for downloading a PDP11. This protocol will permit the transfer of files and mail between the above mentioned machines, and is also the basis for the UNIX access to the Dover.

A subnet gateway has been implemented and placed in operation between the local Ethernet, the Version 1 ringnet, and the Xerox Ethernet. This gateway is in regular use, providing communication between the 11/70 and the Dover spooler, and between the VAX and machines on the Chaos net. Measurements over a recent 24-hour period indicated a total traffic through the gateway of approximately 14,000 packets.

As part of this project, it has been necessary to develop a number of specialized tools, including a fairly sophisticated workbench on the UNIX system for the creation of programs for stand alone PDP-11s. Software now exists which allows us to combine programs written in assembly language, C, and BCPL, all languages which have been used to write programs which we needed to import.

Several slightly longer range projects have also been completed:

- Jerry Saltzer has written two memos outlining a possible approach to networking an environment such as the entire MIT campus.

- Hal Peterson did a study of the congestion control mechanism currently implemented in TCP, a study which indicated certain potential problems with this area of the protocol.

- Kirpal Khalsa completed a preliminary study of specialized flow control algorithms for file transfer protocols.

Several bottlenecks remain, most notably getting XX on the local net and connecting the local net to the ARPANET. ARPA has agreed to deliver an additional IMP to solve this latter problem, and we are importing a Port Expander as a short-range solution.

3. XEROX UNIVERSITY GRANT

During this reporting year the Xerox Corporation, stimulated by proposals from the Xerox Palo Alto Research Center, initiated a university grant program that supplied M.I.T., Stanford, and Carnegie-Mellon University each with 18 "Alto" personal computers, a "Dover" laser-driven xerographic output printer, an Alto-based file storage system, an ETHERNET local network, and a large quantity of supporting interactive software. Installation of most of the equipment was completed by February, 1980, and bridges between the ETHERNET and the other local networks were rapidly developed to allow access to the Dover printer from other computer systems.

The initial use of this equipment has been largely explorational, based on the supplied software, which among other things provides advanced word processing and illustration facilities. The impact of just these facilities, together with the Dover, was clearly noticeable during the Spring thesis season. Quite a number of recent theses, technical reports, and papers have been prepared with this equipment, and nearly all text processing output of the two 545 Technology Square laboratories now is printed on the Dover, which is consuming 150,000 sheets of paper per month. A noticeable increase in the number of illustrations, drawings, and graphs in reports and memos seems to have accompanied use of the Alto report preparation software.

A substantial library of computer games has migrated from other Alto sites. As one might expect, these games are taking a certain toll in graduate student time and attention, although they turn out to be a less serious hazard to academic and research interests than one might expect. Instead, since many of these games demand rapid interaction, they also reveal limitations and requirements for highly interactive software, and on the whole are probably a cultural benefit of the grant. In a similar way, the use of the other software systems is providing both a feel for the depth of engineering required to create a good human interface and an inspiration for some enterprising activists to do better in local implementations of some of the same ideas.

Primarily because of the current availability of LISP machines and expected imminent availability of nu computers, enthusiasm for starting major new programming projects in the Alto environment has been quite low. The programming projects that have started are limited in scope or special in nature:

1) David Reed and Liba Svobodova are supervising design and implementation of the Swallow distributed data storage service on an Alto that can be equipped with several hundred megabytes of disk storage. The research goal of this project is described elsewhere in this annual report. The primary reason for use of the Alto environment is immediate availability of both disk hardware and the Mesa programming system, together with an estimate that the initial implementation will fit easily in the Alto memory space.

2) David Clark has implemented Internet and associated file transfer protocols for the Alto, to allow communication between the Xerox grant equipment and the other computers in the laboratory and the ARPANET community. In conjunction with these protocols, he has deployed a Dover printing service.

3) Robert Schiffenbauer is developing a Mesa-based subsystem for debugging distributed applications.

4) John Guttag is supervising the programming (in Mesa) from formal specifications of a Bravo-like display interface. The purpose is to understand better the implications for programming and system design of working top-down with formal specificatons.

During the coming year a few more research projects are expected to begin using this equipment: a programming specification verification system, some VLSI circuit design work using the ICARUS system, and a bootstrapped CLU compiler have all been discussed.

4. THE *SWALLOW* DISTRIBUTED DATA STORAGE SYSTEM

4.1. Overview

The Swallow project was begun last summer. Its purpose is to design and implement a coherent organization for long-term storage in a network of computers. We assume that these computers are managed in a decentralized way, preserving for each computer in the network a high degree of autonomy. In particular, we would like to obviate any need for a central authority (human or computer) that has complete control of the activities and data in the network. Thus, unlike traditional computer operating systems in which the supervisor manages all computational and memory resources, our distributed environment is much more like a loose coalition

of computers that frequently need to cooperate and to share information, but which computers control completely how they cooperate.

In this context, Swallow can be viewed as a set of standard protocols that cooperating computers may use to manage their data. If Swallow is to be successful in this environment, it must both provide benefits when used and not compromise the autonomy (or course, it must compromise autonomy to the extent of requiring certain standard interfaces).

The benefits Swallow provides include the following:

- *Uniform interface* - the read and write operations provided to users of Swallow make the location of data stored in the system transparent. The owners of data are allowed to control the location of data, however.

- *Reliability* - Swallow provides storage for data objects that is extremely stable. In addition, only those nodes that hold data needed by a computation need be available to run that computation, so availability is enhanced.

- *Atomic Actions* - Swallow provides synchronization and recovery mechanisms so that any arbitrary set of accesses may be combined into an atomic action, using the model developed by Reed [8,9]. Network failures and node crashes do not compromise proper synchronization and recovery of these atomic actions.

- *Protection* - a standard mechanism for encryption-based protection of data stored in the system will be provided. This mechanism is decentralized, so that there is no critical central authority that can compromise the security of every user of Swallow.

- *Support for "small" objects* - novel organizations of storage are needed to support the object model proposed by Reed; at the same time, such storage organizations can be designed to support small objects effectively. The user of Swallow sees an environment consisting of a large number of objects whose average size is relatively small.

These properties are synergistic. For example, in a traditional file system, it is usually not possible to perform atomic actions that involve multiple files. Consequently, objects accessed within the same atomic actions must be stored in the same file. This is one reason that files are large. In the Swallow system, since atomic actions may access multiple objects, it is quite reasonable to store "files" as structures consisting of many individual objects.

4.2. Overall Structure of the Swallow System

Figure 1 illustrates the overall structure of the Swallow system. Each client computer that uses Swallow accesses storage via a module called the broker, which is implemented on each client. The data owned by that computer is stored either on local secondary storage or on a shared server called a *repository*.

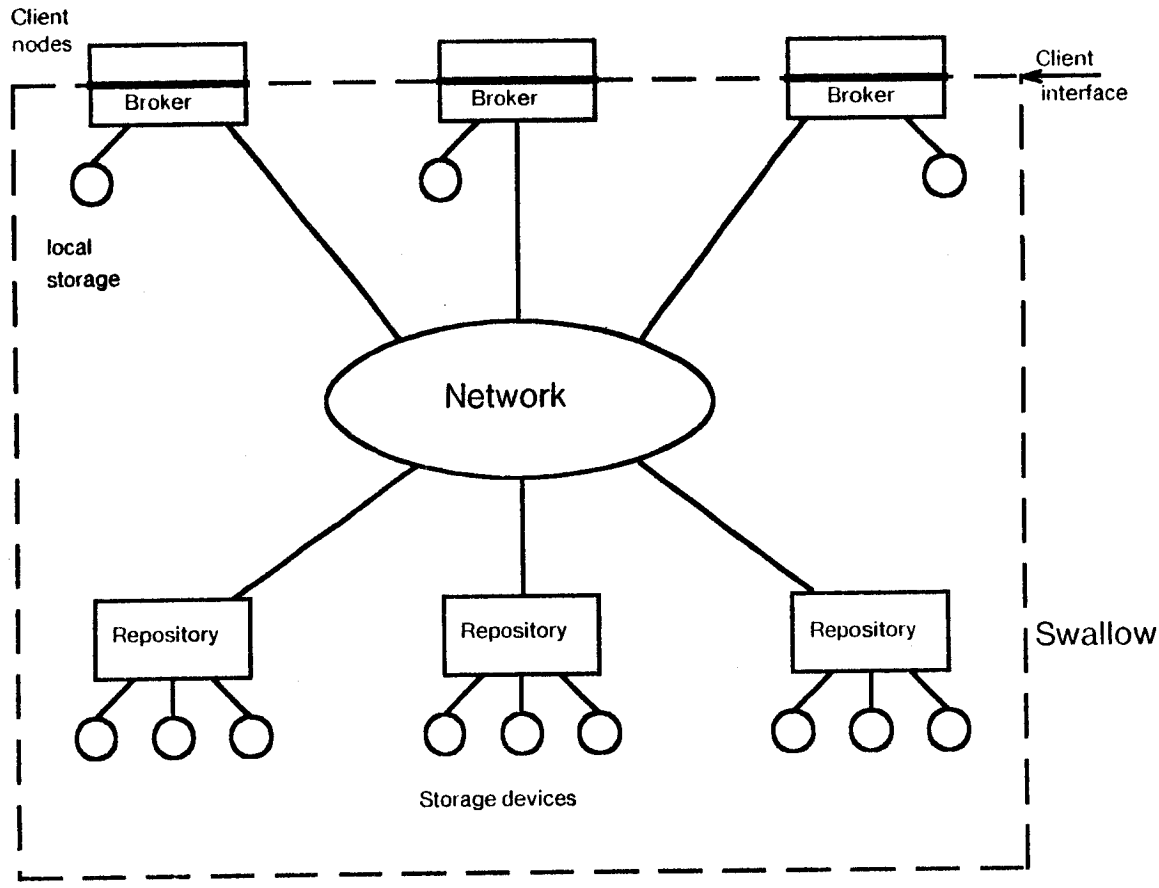

Figure 1: Swallow System Structure

The broker has two functions--it controls the location of, and mediates all accesses to, data owned by its client.

The repository provides large quantities of stable storage. To simplify the job of the repository, a repository is not responsible for protecting the data stored there from unauthorized release.

Both the brokers and the repositories support the protocols needed to provide atomic actions, since both types of modules contain objects that may be used by atomic actions.

4.3. Implementation

We are implementing Swallow to show that the concepts involved (uniform interface, atomic actions, ...) can be implemented in a practical system. Our primary concerns are efficiency and usability. Since the organization of Swallow is radically different from traditional storage systems, the only way to understand how well it will perform in practice is to build it, and then use it in constructing some applications.

Our goal is to implement a prototype system with most of the features of Swallow on a set of Altos, with at least one repository node, and several brokers/client nodes. Altos were chosen because of the existence of both solid hardware and well-developed support software (e.g., Mesa). As the Nu machines become available, we will migrate the system onto the Nu's, first constructing a broker for the ECLU environment on the Nu's, and eventually constructing Nu-based repositories.

4.4. Status

Our efforts for the past year have been aimed at creating an implementable design. The first design phase is nearly complete--we have the following pieces to build.

- *message protocol* supports datagram service for messages of arbitrary length.
- *object access protocol* coordinates interactions between brokers and repositories.
- *version storage management* manages secondary storage (magnetic or optical disk and tape) used for holding the versions of objects.
- *object history manager* maintains the history of versions of objects. Implements stable storage and recovery mechanisms for crashes.
- *repository control* supervises execution of transactions on the repository.
- *commit record manager* implements the two-phase commit protocol among repository and client nodes.
- *broker control* keeps track of objects owned by the broker.

The interfaces and algorithms for these modules have been developed over the past year. During the coming summer, we plan to implement these.

4.5. Major achievements this year

G. Arens has defined the object access protocol used between brokers and repositories. This protocol is "connectionless", that is, the only state information maintained at each node is the values of objects. Since there is no connection state, there is no delay in initiating communications.

L. Svobodova has designed the structure of the repository that supports the object model developed by D. Reed. A write-once storage model (an "infinite tape" with random access) is used to support stable storage of versions of small objects. Techniques similar to real-time garbage collection are used to keep the current versions of frequently used objects in the online portion of the version storage. Emphasis is placed on high performance, particularly on reducing delays due to disk/optical disk latencies. A critical concern addressed in the design is a reconstruction of object histories after a processor crash and recovery from disk failures.

D. Reed has developed an approach to protecting objects using encryption. Objects stored on the repositories are encrypted, with keys known to the owning brokers only. Thus there is no need for implementing a common access policy on the repositories. This simplifies the repositories, and allows clients flexibility to implement arbitrary access control policies.

5. THE AUTHENTICATION SERVER

The authentication server project has two goals. The first is to build a key distribution center that can be used to support other distributed system components that are to be built here. In particular, the Swallow system, described above, will store data in an encrypted form and will therefore require such a server. Also, any secure conversation between processes in the system might require similar services. The main function of the authentication server will be to provide for key distribution.

A second purpose of the project has been to provide a source of experience with programming for a distributed system. The currently available "extended" CLU has served as the language for several experimental implementations. By reviewing our programming experiences regularly, we are developing some insight into how such a language can support the implementation of programs for a distributed environment.

We began meeting in September 1979, before any other projects had developed detailed specifications of their authentication server requirements. We spent about two months reading selections from the literature on protection and encryption, as well as learning "extended" CLU. At that time we decided to implement the

protocols for establishing a secure conversation as presented in. There are two versions of the protocols, the first for use with conventional encryption and the second based on public key encryption. These protocols should have some relationship to protocols required by local users, but are not particularly tailored to the needs of other projects in the laboratory. Thus the main results of this exercise have been initiation into the extended CLU programming environment, production of two simple servers that can serve as foundations upon which to build, and identification of a variety of problems not addressed in the Needham and Schroeder paper.

The next phase of the work was to redesign the programs so that communication could proceed in terms of internal datagrams. The group defined a datagrams standard for use with the Needham-Schroeder protocols [3] and have begun a new implementation.

There are three kinds of future work that we are considering. First, there are still parts of the current implementation of the Needham and Schroeder work that are incomplete. The encryption procedures do not implement secure encryption algorithms. Also, Needham and Schroeder suggest some modifications to their protocols that would facilitate caching of keys for reuse in future conversations. The current implementations require that the authentication server be involved each time a new conversation is started.

Second, there are issues that were outside the scope of the Needham and Schroeder paper that we can tackle. These include protocols for proceeding with a conversation once a key has been agreed upon and protocols for revoking a key once it has been compromised.

Third, we are interested in providing services that will be of use to people building other programs. For example, if the data storage server provides storage for large numbers of small objects, each under a separate key, then adequate performance may depend on its ability to get a large number of keys from the authentication server in response to a single request.

6. APPLICATIONS FOR DISTRIBUTED SYSTEMS

6.1. The Application

In the area of applications we have continued to focus on distributed calendar systems. There are two kinds of calendars that we have been designing -- personal calendars and public "resource scheduling" calendars.

The personal calendar can be used for keeping track of appointments, meetings, holidays, etc. The calendar can be displayed in several ways showing either a summary of the week, a list of appointments on a day, or a diagram of the day showing blocks of free and reserved time. The main operations are "appt" to make an appointment, "cancel" to cancel one, and various display commands. One can attempt to make an appointment at any time. If there is a conflict with another appointment, the calendar reports this fact. If not, the appointment will be made. Appointments are recorded at a particular time with a few keywords to indicate the purpose.

The Conference Room Calendar is similar to the personal calendar in that time slots can be reserved and cancelled. This program is meant to support the reserving of time in one of our conference rooms in the laboratory. The room is generally used for seminars and may involve the coordination of several people and resources. Since a seminar generally has a host who is responsible for the reservation, the host's name is listed in the calendar display as the keyword for the appointment. In addition, there is a form on file for each appointment. The form contains information about the seminar such as the speaker's name, the title of his talk and whether there will be refreshments. These forms can be active, in which case they may trigger communication with other calendars (such as, the calendar for the person who sets up the coffee pot in time for scheduled refreshments).

6.2. Meetings

A personal calendar can try to call a meeting. The desired length of the meeting, a set of possible times and a list of participants must be specified in the request. The calendar system will try to find a time at which the meeting can be held and will then notify all participants.

For meetings that are called very far in advance of the time at which they will be held, the meeting can be considered to be tentatively scheduled. A scheduler will keep track of several possible times at which the meeting can be held. A second meeting is considered to conflict only if scheduling it (and therefore, removing its time slot from the set of times tentatively reserved for the original meeting) would reduce the set of possible times to less than one. If the second meeting is scheduled, the set of available times for the first meeting is simply reduced. Shortly before the date of the meeting a single time is chosen for the meeting. This can occur either at a "commit" time specified in the call for the meeting or by an explicit request to commit. A caller could specify that he wants a meeting the week of March 10th and that it should be definitely scheduled by March 3rd. Thus the caller can be sure that the meeting will appear on his calendar with sufficient advance notice for

planning. If the meeting is committed to a single time too soon, it is quite likely that some participant will have to cancel in order to meet a higher priority commitment that arises later. This would require rescheduling, rather than the simple reduction in the set of tentative times.

6.3. Calendars in a Distributed System

Facilities for coordinating a set of calendars are of use in either a centralized or a distributed system. If the system is to be distributed, its implementation will certainly differ from the implementation of a centralized version. We are assuming that in order to coordinate with another calendar a request must be sent to that calendar. That is, there is no central data base that contains information on all calendars and that can be accessed directly by any calendar.

Operations other than calls for meetings may depend on data at more than one node. For example, when there are tentative meetings (as described in section 2.3) then while a meeting is "uncommitted" the status of certain time slots on the personal calendars of the participants may depend on the status of the tentative meeting. Thus even if the personal calendars store their data locally they may have to communicate with the tentative meeting in order to find out whether a particular time slot is free. This can cause noticeable delays if a user is at the terminal trying to schedule an appointment in real time. It also raises a question as to how to display the calendar--should all tentative times for various meetings be shown or should the display show a possible schedule based on information available locally?

Other questions arise:

- How do these data dependencies relate to the dependencies which arise in supporting modular atomic transactions [8]? Are such dependencies at the application level likely to occur in many applications? If so, how can we support their implementation in a programming language for distributed applications?

- Should the caller of the meeting act as the source of information about the tentative meeting? If the tentative meetings are distributed in this way how will scheduling be done if one person is invited to several meetings? Should a central scheduler be invoked to manage meetings? (This latter approach is being explored by a UROP student.)

- Should chains of tentative meetings be schedulable? (E.g., Can I schedule Meeting A conditionally depending on the final timing of

Meeting B?) This may save the time of checking with the tentative meeting about a particular time slot. But then how will the system help me in backing out of meetings when conflicts are later confirmed?

6.4. Progress

We have implemented several versions of the calendar. A working version of a single user is available on XX. Draft descriptions of the calendars have been proposed in an internal working paper. A first version of tentative meetings in multi-user calendar system has almost been completed by a UROP student, Pat O'Donnell. The user interface has been studied and a version implemented by a Bachelor's Thesis student, Eli Wylen [7].

7. MISCELLANEOUS

7.1. Research in Object Oriented Systems

We have claimed that effective development of distributed system semantics is strongly enhanced by the object oriented view of systems and languages; the view that makes the language or system directly aware of the potentially small storage units which hold the individual data items of relevance to the programmer. Allen Luniewski, in a Ph.D. thesis, has explored a machine architecture which directly supports this small object view of data management. His thesis suggests that it is possible to provide a reasonable implementation of an object oriented machine, in a manner independent of a particular programming language. In particular, he has demonstrated an architecture that potentially permits objects defined in different languages to be exchanged. In particular, compile time typesafe languages and runtime typesafe languages could presumably coexist in his environment.

7.2. Miscellaneous Distributed System Techniques

Andy Mendelsohn has been investigating the distributed implementation of interactive programs. One example is a distributed editor, with the functions of the editor distributed between a "front-end" personal compiler with highly interactive input and output and a "back-end" compiler such as a timesharing system with higher performance and more storage. The goal is to develop general techniques for distributing functions in any application between a highly innovative front-end and the other compilers in the network. The major accomplishment this year has been the design of a distributed text buffer.

References

1. Daniels, D., Lucassen, J., and Rubin, W., "The authentication server: a datagram implementation," (Draft) May 22, 1980.

2. Kent, S. T., "Encryption-based protection protocols for interactive user-computer communication," MIT/LCS/TR-162, MIT, Laboratory for Computer Science, Cambridge, MA, May, 1976, pp. 121.

3. Needham, R. M. and Schroeder, M. D., "Using encryption for authentication in large networks of computer," Communications of the ACM 21, 12, December, 1978. pp. 993-997.

4. Gifford, D. K., "Weighted voting for replicated data," Proceedings of the Seventh Symposium on Operating Systems Principles, Pacific Grove, CA, December, 1979, pp. 150-162.

5. Reed, D. P., "Implementing atomic actions on decentralized data," Preprints for the Seventh Symposium on Operating Systems Principles, Pacific Grove, CA, December, 1979, pp. 66-74.

6. Reed, D.P., "Naming and synchronization in a decentralized computer system," MIT/LCS/TR-205, MIT, Laboratory for Computer Science, Cambridge, MA, September, 1978.

7. Wylen, E., "A personal calendar: the human-computer interface," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

Publications

1. Corbató, F.J. and C.T. Clingen, "A managerial view of Multics system development," reprinted in Tutorial: Software Management, Donald J. Reifer (Ed.), IEEE Chapter Society, 1979.

2. Luniewski, A., "The architecture of an object based personal computer," MIT/LCS/TR-232, MIT, Laboratory for Computer Science, Cambridge, MA, January 1980.

3. Marcum, A., "A manager for named, permanent object," MIT/LCS/TM-162, MIT, Laboratory for Computer Science, Cambridge, MA, April 1980.

4. Reed, D., "Implementing atomic actions on decentralized data," to be published in Communications of the ACM.

5. Saltzer, J., "Environment considerations for campus-wide networks," Internet Note No. 143, March, 1980.

6. Saltzer, J., "Source routing for campus-wide internet transport," Internet Note No. 144, March, 1980.

7. Sollins, K., "The TFTP specification," Internet Note No. 133, January 1980.

8. Stark, E., "Semaphore primitives and starvation--free mutual exclusion," MIT/LCS/TM-158, MIT, Laboratory for Computer Science, Cambridge, MA, March 1980.

9. Svobodova, L. and D. Clark, "Design of distributed systems supporting local autonomy," IEEE COMPSON Spring '80, February 1980, invited paper, 438-444.

Theses Completed

1. Ames, W., "A local area network simulator," M.S. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, September 1979.

2. Finseth, C., "Theory and practice of text editors," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

3. Goldberg, D., "A character oriented display editor," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1980.

4. Khalsa, K., "Flow control algorithms for file transfer protocols," S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

5. Krueger, S., "System features to aid in the on-line diagnosis of computer peripherals," S.M. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980 (also B.S. degree).

6. Leckband, C., "A design of reliability mechanisms for defense minicomputer systems," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

7. Luniewski, A., "The architecture of an object based personal computer," Ph.D. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1980.

8. Myers, B., "Displaying data structures for interactive debugging," E.E. and S.M. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1980.

9. Peterson, H., "Design of source quench congestion control algorithms in interconnected networks," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

10. Pettinato, S., "A sports scheduling system," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

11. Reuveni, A., "The event based language and its multiple processor implementations," Ph.D. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, November 1979.

12. Seaquist, C., "Semantics of synchronization," Ph.D. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

13. Smith, M., "An internet implementation of a terminal access protocol for multics," S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

14. Stark, E., "Semaphore primitives and starvation-free mutual exclusion," Ph.D. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1980.

15. Stathis, G., "A computer controlled telephone dialer," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1980.

16. Toner, S., "Dynamic message routing in interconnected local area data networks," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, August 1979.

17. Vieraitis, R., "Evaluation of the performance of a local area network," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

18. Wylen, E., "A personal calendar: the human-computer interface," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1980.

Theses in Progress

1. Arens, G., "Recovery of a repository in a distributed data storage system," M.S. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1980.

2. Baldwin, R., "An evaluation of the recursive machine architecture," M.S. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected June 1980.

3. Kent, S., "Implementing external protected subsystems in small computers," Ph.D. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1980.

4. Ketelboeter, V., "Foward recovery in distributed systems," S.M. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1980.

5. Mendelsohn, A., "Tools for building user interfaces to distributed systems," S.M. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1980.

6. Schiffenbauer, R., "Debugging in a distributed system," S.M. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1980.

7. Simpson, G., "A monitoring station for a local area network," S.M. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected November 1980 (also B.S. degree).

8. Thomas, J., "A mult-protocol network mail transport facility for Multics," S.B. thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1980.

Conference Participation

1. Corbató, F.J., Awarded Harry Goode Award, at the National Computer Conference, Anaheim, CA, May 19-22, 1980.

2. Reed, D., "Implementing atomic actions on decentralized data," ACM Seventh Symposium on Operating Systems Principles, Asilomar, CA, December 1979.

3. Saltzer, J., Invited Commentary on Distributed Systems," ACM Seventh Symposium on Operating Systems Principles, Asilomar, CA, December 1979.

4. Saltzer, J., Summer Study on Air Force Computer Security, Draper Laboratory, Cambridge, MA, June/July 1979.

5. Svobodova, L., "Reliable distributed systems," IFIP Working Conference on Reliable Computing and Fault-Tolerance in the 1980's, London, England, September 1979.

Lectures

1. Chiappa, N., "The MIT LCS network," talk presented to the Data Communications Group of London of the British Computer Society, University College London, London, England, September 17, 1980.

2. Clark, D., "Local area networks," Greater Boston Chapter of the ACM, Boston, MA, April 24, 1980.

3. Corbató, F.J., "An overview of computer science research at MIT," MIT-ILP program for several companies, Tokyo, Japan, July 9, 1979. Also presented at Tsinghue University, Peking, China, June 27, 1979.

4. Corbató, F.J., "A management view of the Multics system development," MIT-ILP program for NTTPC, Tokyo, Japan, July 10, 1979.

5. Kent, S., "Implementing external protected subsystems in small computers," IBM San Jose Research Laboratory, San Jose, CA, March 24, 1980.

6. Mendelsohn, A., "Tools for building user interfaces in a distributed processing environment," Hewlett-Packard Computer Laboratory, Palo Alto, CA, April 23, 1980.

7. Reed, D., "Using naming for synchronizing access to decentralized data," University of Rochester, Rochester, NY, October, 1979.

8. Reed, D., "Implementing atomic actions on decentralized data," Digital Equipment Corporation Research, Maynard, MA, November 1979.

9. Reed, D., "The distributed data storage system," IBM San Jose Research Laboratory, San Jose, CA, December 1979.

10. Saltzer, J., "The impact of modern technology on system design," a series of five lectures given at: Indian Institute of Technology, Delhi, January 16, 1980; Indian Institue of Technology, Kanpur, January 18, 1980; Indian Institute of Technology, Madras, January 30, 1980; Indian Institute of Technology, Bangalore, January 29, 1980; Computing Society of India, Hyderabad, January 31, 1980.

11. Saltzer, J., "Workshop on distributed systems," organizer and lecturer, President Hotl, Bombay, India, January 21-25, 1980.

12. Svobodova, L., "Operating systems for distributed computing," Technical Vitality Program, State University of New York, Binghamton, NY, November 1979.

13. Svobodova, L., "Modeling and semantics of distributed computation," Technical Vitality Program, State University of New York, Binghamton, NY, November 1979.

14. Svobodova, L., "Distributed storage system for a local network," PRIME Computers, Framingham, MA, June 1980.

Committee Memberships

Chiappa, Noel, DARPA IPTO Internet TCP Working Group

Clark, David, DARPA IPTO Internet TCP Working Group

Greif, Irene, Program Committee for Principles of Programing Languages

Reed, David, DARPA IPTO Internet TCP Working Group

Saltzer, Jerome, Draper Laboratory Committee on 1979 Security Workshop

Saltzer, Jerome, DoD/DDRE Security Working Group Member