LOCAL NETWORKING AT THE M.I.T. COMPUTER LABORATORIES:

A User's View

Marek W. Lugowski

Indiana University Computer Science Department

Technical Report #173

November 1984 - June 1985

Abstract:

The user's view of the computing environment at the MIT Artificial Intelligence Laboratory and the MIT Laboratory for Computer Science is presented with emphasis on the local area network's performance, history, and configuration. The author's year-long engagement as an AI Lab staff member and network user is used as a basis for reporting on the over-all performance and usefulness of the MIT local area networks. Some technical information, that contained in the MIT-published network specifications and that which has never been written down, is presented. A short analysis of the merits of the MIT computing environment design and implementation is provided, to the effect that the networked terminal traffic and the file-transfer traffic should not be routed simultaneously over the same network links.

Keywords:

Local base-band area networks, time-sharing terminal, file-transfer protocols, design of distributed computing environments, lisp machine, history of artificial intelligence, office systems

1. Environment

The computing environment described in this paper is an important subset of MIT's total computing environment, and it consists of The Artificial Intelligence Laboratory (henceforth, AI Lab), The Laboratory for Computer Science (LCS), the largest mainframes in the Department of Electrical Engineering and Computer Science (EECS), and Symbolics, the nearby upstart company launched by the one-time MIT "moby wizards" to develop and manufacture Lisp Machines. Lisp Machines were invented at MIT by, among others, the people who are now at Symbolics, and one could say that Symbolics and MIT enjoy a close symbiotic relationship, to the point of sharing the same local area network. This uncommon--in the academia--intermingling of the university with the corporate is characteristic of MIT. The LCS and the AI Lab, for example, share their building with Polaroid, and a third party still owns and operates the premises.

The organizational checkerboard is closely mirrored by the wealth of different equipment to be found in the vast joint machine room of the LCS and the AI Lab. One will find there the largest DEC-20s in the world, the KL-10Bs with numbing numbers of disk drives hanging from them, side by side with ancient, truly core-memoried KI-10s with huge banks of custom-build memory (256K), next to the newest personal computer equivalents of Ferraris, the Symbolics 3600 Lisp Machines ($100K). There are also ancient home-built lisp machines (so-called CADRs), numerous VAX 11/780s and a something called "the VAX farm": 50 or so 11/750s, side-by-side, most deprived of their VT125 consoles, which goes to show that at MIT it's easier to get hold of a VAX than a terminal.

All these pieces of equipment have to be connected to each other, as well as to a myriad of special devices such as the Alto-driven Dover, a huge Xerox laser printer that is the main hard copy device for both labs. Then, there is a collection of odd peripherals such as robot arms, cameras, digitizing equipment of all sorts, the Arpanet gateway's BBN computers (IMPs), as well as scores of lesser toys.

Each one of the above, including the mainframes at the EECS some third of a mile away, is a node on the local area network known as Chaosnet, or on a similar local area network consisting of Ethernet hardware running TCP/IP protocols. The LCS owns and maintains all of the TCP/IP cable, but more than 50% of its local area networks are Chaosnet links. All of the AI Lab is connected to the Chaosnet. Aside from computers, the same applies to the hundred-plus terminals, mostly Ann Arbor Ambassadors, that are sprinkled all over the AI Lab, and to their LSI-11 concentrators. The Symbolics part of Chaosnet is connected to 545 Technology Square (the MIT computer science labs) via a roof-mounted microwave link.

2. Inventing equipment ahead of the marketplace

A good deal of the above environment's hodge-podge character owes itself to the MIT AI Lab's penchant for designing and constructing equipment that only later becomes commercially available. The net result of this activity is, of course, a great contribution to computer technology in general, but also a chaotic, home-made look of the equipment in the AI Lab's machine room. In fact, quite often there may be around only one or two people qualified to maintain a particularly obscure device, and certainly no outside repairman would be willing to mess with this hardware. The home-made core memories, the home-made lisp machines, and the home-made peripheral devices, as well as the bridges connecting Chaos links, all fit in this category.

3. Problems peculiar to the MIT computer labs

The labs have a long tradition of using the DEC PDP-10 computers. Each lab has a huge DecSystem-20 and a couple smaller KI-10 systems, some almost on the way out. The hackers at the labs have been historically fond of these machines, and a lot of them refuse to hack on anything else but those (and, of course, the lisp machines).

The sentiment for using these machines is strong because of popular acclaim for the Incompatible Timesharing System (ITS), an operating system developed at MIT and preceding TOPS-10/20, Unix, and VMS. The ITS lovers claim that a lot of now-standard operating system concepts were pioneered under ITS, and that the commercial products mentioned above contain inferior spin-offs (or downright faulty implementations) of the ITS mechanisms. ITS owes its name to the fact that it heavily depends on the PDP-10 architecture, mainly for efficiency reasons. It is not used anywhere else in the world--a fact that does not faze the ITS hacker one bit.

While the older PDP-10s in the labs run ITS, the newer ones run the modified DEC TOPS-20 operating system. The MIT sites found it so lacking that a great deal of it was modified to accommodate local area networking and improve the user interface. Even now, after years of trying, the AI Lab's main machine, the Oz, remains off the Arpanet because the TOPS-20 operating system cannot adequately handle all the network traffic involved in being on a local net and on the Arpanet, and having the bulk of its terminals be network-wired as well. The operating system seems to simply run out of buffer space.

When failures of commercial products occur in the highly partisan

research environments like MIT AI Lab, the general tendency is to hang onto old home-grown standbys and do without. It now seems that about the only reasons the Oz is continuing to run TWENEX (modified TOPS-20) are the superior protection scheme (under ITS it was considered downright unfriendly to protect one's files in any way; snooping was part of the etiquette), the convenience of the newer users who missed becoming initiated into ITS, and the ability to have file names longer than 6 characters (ITS efficiency at work!).

The LCS, on the other hand, has no compunction about conforming to industrial standards. While the AI Lab went off to develop its own local area network ideas (Chaosnet), the LCS happily installed TCP/IP on prototypical hardware from Xerox (Ethernet). While the AI Lab continues to boil over what operating systems ought to be supported in addition to the ITS, the LCS cheerfully prepares to eliminate everything but the Lisp Machine (Symbolics and the TI-made LMI model), and Berkeley 4.2bsd. The LCS is in the process of putting a separate Ethernet in its part of the building in order to accommodate a bit-mapped display (of Apple Lisa class) in every one of its offices. The LCS mangagement is also considering a Symbolics 3600-class machine for every office, likely the Texas Instruments' The Explorer.

Despite these differences, the two labs are condemned to share the same facilities for output, the same machine room, and--of course--the same aggregate of local area networks. The joint computer labs at MIT may well be the most exasperating territory for a systems designer and a systems programmer to work at.

4. Chaosnet, MIT's own local area network architecture

a) INTRODUCTION

Before the Ethernet was commercially available, the people at the MIT computer science laboratories decided to develop and install their own local area network. The need was motivated by the parallel effort at developing the lisp machine. Lisp machine, conceptually a multi-processor, was to comprise a set of dedicated CPUs, memories, and swapping disks, but with a central file system. The dedicated CPU, disk, and memory were provided in order to assure a constant level of high quality performance. The shared file system was designed to facilitate centralized back-up and maintenance, as well as to provide mail interaction, the ability to send messages interactively, and the sharing of programs. The Chaosnet was to be the file-transfer medium and the communications bus of this system.

The specification to the effect that Chaosnet was to the file system of the lisp machines implied a design with fast response and throughput, great reliability, and one that would make it possible to connect maybe a hundred of hosts. Unlike the Arpanet (which was already in existence then), there was no need for operating over long distances. Chaosnet would also be used to access other shared resources like tape drives, printers, and various specialized peri-pherals. The above guaranteed an environment consisting of many different machines; thus the Chaosnet implementation would need to be simple in order to accommodate this variety and in order to provide for effective maintenance. The ability to isolate a network failure is a direct consequence of the simplicity of the network and its interface to hosts.

The high performance aspects of the design included using a very high speed transmission medium, and operating it in a simple, low-overhead fashion, as opposed to the employment of particularly clever algorithms. (Of course, too simple an algorithm could have very well badly impaired the performance of the net; nevertheless the basic tenet of simplicity as antecedent of high performance squares nicely with simplicity as antecedent of tractable maintainability.)

Aside from ignoring the long distance question, Chaosnet design did not address having low-speed links, high error-rate links, multiple (redundant) paths, or multiple levels of service, or even secure communication, other than end-to-end encryption done by the hosts.

While predating the commercial availability of the Ethernet, the Chaosnet project heavily borrowed on the research already done at Xerox PARC, as well on the concepts developed in TCP and the Arpanet. Nowadays, a lot of the hardware on which the Chaosnet runs is the very same thing supplied by the Ethernet vendor.

b) HARDWARE

The physical medium of the Chaosnet is the 75-ohm TV cable. This cable is connected to a set of nodes. A node consists of a host running a program called the Network Control Program (NCP), an interface connected to the host's I/O bus, and a 10-meter flat cable connecting the interface to a cable transceiver, which is just a tap on the main cable. The NCP manages and controls the network. Note that this means an absence of centralized control.

Essentially, a node may seize control of the cable and transmit a packet. This packet arrives at all other nodes on the cable. This also implies (in the absence of customizations in hardware) a ban on

cable branches, stubs, or circular topology. Also, each node decides whether it accepts the packet or not.

DC attentuation and dispersion dictate that the length of the cable not exceed 1 kilometer (0.62 mile). This distance could be exceeded through the use of amplifiers, but in practice this is not done. Instead, bridges are used. A bridge is a PDP-11 (usually, an LSI-11) computer relaying packets from one network interface to one or more others. These bridges also act as terminal concentrators, and they run a special program for that purpose called MINITS. It allows the users to choose from a menu of cable hosts, and then attach to any one of them. This is the mechanism in which most of the AI Lab's terminals are network-connected, as mentioned before.

Other networks are attached to the bridges, as well. This is how the Chaosnet talks to both the LCS's TCP/IP and the BBN IMPs running Arpanet. Also, the PDP-10s are connected via DEC's high-speed computer-to-computer interface to the bridges rather than through the node set-up described above.

The packet is a sequence of up to 4032 bits plus 48 bits of header. Packets are grouped into 16-bit words. The packet set-up facilitates error control and cable allocation. The entire point of Chaosnet's hardware is to deliver these packets from one place to another. The software protocols resident in NCP make sure to manage this service, compensate for hardware failures, and generally provide more useful services than simple transfer of packets from one computer to another.

The packet's header is 3 16-bit words: destination, source, and circular redundancy check. The source may not be the original sender of the message, but merely the node which posted the packet on this

cable, since the message might have originated on a different cable. Similar considerations apply to the destination word.

The cable transceiver, or the tap, is a small box connected to the cable via a UHF connector and a T-joint. It contains the analog portion of the interface logic, provides the ground isolation between the cable and the host, and contains some protective circuitry that would (hopefully) prevent a buggy NCP program or interface from jamming the cable continuously. This aspect of the transceiver could be much improved, judging from past failures. The transceiver receives a digital signal from its host and puts it on the cable as a level of 8 volts for binary 1 and 0 volts (open circuit) for binary 0. It uses a very fast VMOS power FET. When the cable is idle, it is held at 0 volts by the terminators, which is just the opposite from the ISO standard for synchronized transmission. The transceiver compares the cable against a reference voltage, and returns the differential signal to the interface. In addition, it detects interference (or packet collision) and informs the interface about it. The transceiver has LEDs for power OK, transmitted data, received data, and packet collision. It also has a test button to emit a continuous stream of binary 1s as if supplied by the interface. These features have been proven quite useful in quickly tracking down network problems.

The interface is a wire-wrap board containing some 120 TTL chips. It implements the network's hardware protocols, buffers the packets going both ways, handles the error-checking, and interrupts the host when a packet comes in or out of the buffers. The interface provides packet synchronization so that the host may generate packets asynchronously. There are interfaces for lisp machines, LSI-11's, and the DEC Unibus. The Unibus covers all of the other interfacing needs if something doesn't have a Unibus, it is connected to a bridge.

c) HARDWARE PROTOCOLS

These protocols deliver packets from node to node on the same cable. They provide a decent probability of successful delivery and packet integrity (i.e., if a packet is compromised, say so or discard it).

Bit representation is Upright Biphase NRZI. It relies on the self-clocking provided by a crystal clock included in each interface. Each bit duration (cell) begins with a stage transition from low to high or vice versa. Thus, the beginning of the cell is marked, and self-clocking is provided. 3/4 through the duration, the state of the cable is sampled: binary 1 is high. If the current bit is the same as the previous bit, there will occur another transition. If it isn't, then there will be no transition since the clock will have set the state correctly at the beginning of the cell.

The AC frequency of the signal on the cable varies between 1/2 the bit rate and the full bit rate. The information bit rate is 4 M bits per second. The self-clocking feature accommodates slight variations in transmission and cable propagation speed. The NRZI scheme guarantees the existence of at least one state transition per duration of bit, i.e., every 250 nanoseconds. Thus, it is possible to make inferences about the cable based on the presence or absence of state transitions.

For example, if the cable remains low for more than 2 bit durations, it is considered not busy. This condition corresponds to the end of packet and thus allows someone else to grab the net. Remember that if no transceivers are active, the cable is held at 0 volts (low).

Conversely, if the cable remains high for 2 bit cells, it signifies an "abort signal". The abort signal is used in two ways. First, if

a transceiver detects a packet collision, the two sending interfaces are made to send an abort signal (1000 nanoseconds), as well as to cease transmission. The abort signal tells the receivers to ignore the packet and insures that the other sender also aborts. Second, the abort signal is used in flow control: When a receiving interface decides that it wants the incoming packet, but its receiving buffer is full, it sends the transmitter an abort signal ordering it to stop.

The packet transmission occurs in reverse bit order, to simplify hardware. Thus, the header words which are transmitted at the end of the packet arrive first, in the order check, source, and destination. The data words, in the reverse order, follow. Words are transmitted least-significant bit first. The software is never aware of the fact that the transmission is reverse order; packets arrive at the their destinations in the order sent. At the end of the packet, an extra binary 0 is added to bring the cable into low state, so that an extra spurious clock transition is not generated when the cable goes idle. This bit, not unlike in the HDLC bit padding, is transparent to the software.

The technique used here is carrier sense: An interface will not start transmitting unless the cable is idle. Thus, collisions may occur only at the beginning of packets. Once a transmission has successfully gotten under way, the cable has been "seized" by the sender, and the transmission will continue successfully to the end of the packet, however long it takes. Also, the amount of time wasted by a packet collision is therefore limited to the round-trip cable propagation delay.

A time-division technique is used to prevent any two interfaces from initiating transmission at the same time. It only costs a small delay in the initiation of transmission, while preventing almost

all collisions. Its trade-offs pay off better as the load on the cable increases. Essentially, each interface is assigned a time-slot called a turn, in accordance with its network address, and may initialize transmission only when its turn has come. The turns are set far enough apart that one interface's cable "seizure" will make the other interfaces see that the cable is busy when their turn arrives. Each interface is provided with a counter to keep track of turns while the net is idle. Each packet synchronizes these counters by setting them from the source address of its sender; at the transmission time, it must have been the sender's turn to transmit. This strategy is analogous to token-passing in ring networks, with the counting being the virtual token.

The virtual token metaphor is useful in analyzing the consequences of the above scheme. The token travels through the net at a much lower speed than that of a packet; at best, when the nodes are very far apart, it travels at just half the speed of a real signal. Also, the Chaosnet has the topology of a line segment, not a circle. Thus, the token is additionally slowed down by having to retrace its way. The slower virtual token speed is the price of Chaosnet's superior robustness to that of ring networks. In reality, the token is slowed down even more to provide a margin of safety. A typical value for the token's round trip time is 64 microseconds. While the Chaosnet designers feel this is perfectly adequate for the network's primary task--that of file transfer--it becomes a real factor in system performance from the viewpoint of a user logged in on a networked terminal, such as the ones at the AI Lab.

Also, if the cable has been idle for a long time, the various clocks will have lost synchronization. If a source address is corrupted, any interface that sees this address will mis-set its clock accordingly.

Additionally, a packet might occasionally collide with a random noise burst rather than another packet. Finally, the sender cannot tell a receiver-busy abort signal from real collisions. There are no randomizing mechanisms in the hardware protocols, since the higher-level software realizes that a packet was lost and retransmits it. The basic assumption in the architecture design of the net is that there is enough randomness in the software that the two colliding nodes will not collide again on the retransmission in a periodic lock.

d) OVERVIEW OF SOFTWARE PROTOCOLS

These protocols arrange for high-speed interchanges between hosts, regardless of location (hosts may reside on different cable links), and without undetected transmission errors. The original design goals called for the speed of file transfers to be comparable to that of inexpensive tape drives, 3 K characters per second, which is about 10 times the speed of file transfers on the Arpanet. The actual transfer speed approaches twice this specification in favorable circumstances.

This success was brought on by designing out the bottlenecks that can be found in the Arpanet, such as the control link needed for the acknowledgement of every message before the next one may be sent. The goal of design simplicity has also played a role: a full Chaosnet NCP is about the half the size of its Arpanet equivalent on the same machine. The Chaosnet's low-performance implementations may omit some features still. There's even a minimal implementation for a single-chip microcomputer.

From the point of any two users, the chief result of running Chaosnet is a connection. A connection is a reliable packet-transmission channel operating in a full-duplex mode. The network promises to

never lose, duplicate, garble, or resequence the packets. Any unrecoverable errors promptly result in breaking a connection and informing the two users. It is up to the user software to either deal with the communication in terms of packets or ignore their boundaries and treat the connection as two half-duplex streams of 8-bit or 16-bit bytes.

File access, interactive terminal connections (remote logins), and handling of data in other byte sizes such as 36 bits are all built on top of the connection facility by the user programs, which from the software protocols' point of view include the higher protocols to be mentioned later.

To start a connection, the two processes must contact each other. In the usual user/server scenario, the server process does not exist initially and must be created and made to execute the appropriate code. The contacting is implemented as follows: One process is deemed the user, the other the server. The server possesses some contact name for which it listens. The user asks its local host's operating system to be connected to the server, specifying the network code and contact name. The local operating system sends a message (a Request for Communication) to the remote operating system (of the server-to-be), which then creates the server process, or explicitly rejects the request. Chaosnet does not deal with the problem of automatic discovering of which host to connect in order to obtain a particular service.

In the case of two existing processes already knowing about each other and wishing to communicate, one is arbitrarily designed as server and one as user. (This is much the same policy as in the IBM SDLC protocol.)

Each node on the network has a unique address, which is a 16-bit number. The most significant 8 bits signify a subnet, and the other 8 bits specify a host within the subnet. A subnet can be either a physical Chaosnet cable or a high-speed host-to-host connection between a bridge and a PDP-10.

Since the Chaosnet was designed with the lisp machine in mind, the data transmitted through it generally follows the Lisp Machine standards. (The Lisp Machine, as a set of standards, was developed exclusively at the MIT AI Lab in the early 1970s, and any company, such as Symbolics or LMI or TI, wishing to produce MIT-based lisp machines must conform to them. They are available to the public.)

Unlike some other networks, the Chaosnet does not do any software checksumming. This is so because the great variety of hosts with different architectures Chaosnet connects makes it prohibitive to attempt implementing an algorithm which can be executed compatibly and fast on all hosts. Instead, all error-checking is done in hardware of interfaces. The assumption is that other types of packet damage not detectable by the interface algorithms will produce results so obvious as to be detected and fixed immediately.

Routing of packets is made on packet-by-packet basis and has nothing in common with the concept of connections. This makes it possible to make the bridge software simple and easy to implement. (Bridges connect different cables of the same network--Chaosnet--and thus, among other things, forward packets.) To prevent loops, each packet contains a forwarding count. Each bridge increments this count as it handles the packet. Should the count ever reach a maximum, the packet is discarded. From there on, it is the error control protocol's decision whether to recover the packet or conclude that the communication link cannot be salvaged at the

current time.

Briefly, the routing works as follows: a packet's destination address is checked by the host. If it is this host, receive the packet. (Intra-host processes may very well communicate via the network.) Otherwise, increment the forwarding count and discard the packet if the count has exceeded its maximum. If the packet's destination lies on the same subnet as the host, transmit the packet on that subnet, else look up the subnet in own routing table to find the best bridge to that subnet, and transmit to that bridge.

The concept of best bridge is implemented using a cost factor that resides in the routing table. The cost for subnets is increased by 1 every 4 seconds, typically doubling after a minute. Each cost has a ceiling which it does not exceed, thus preventing arithmetic overflow. The reason for increasing cost is to discount the value of old information. Directly connected subnets are not subject to this cost increase.

The updating of the routing table happens via each bridge periodically advertising its presence (every 15 seconds) by broadcasting a routing (RUT) packet on all its directly connected subnets. The RUT packet tells its recipients which subnets that particular bridge serves best. As a consequence, the routing table is updated to say that this bridge should be used for that subnet.

When there are equivalent bridges, the traffic is divided among them only through the happy circumstance that their RUTs were sent out at different times. This simple scheme could be supplemented by more sophisticated bridge scheduling on the host level, but this is not done (because who would wish to hack such "hairy" boredom), and as a consequence some subnets can get disproportionately

overloaded at times of "rush hour" traffic, thus severely impeding performance. This is the reason why Chaosnet routing has such importance to the user view of the net.

Flow control and error control are implemented in the NCP through retransmission. Retransmission continues until the receiver sends to the sender a signal called a receipt. A receipt indicates that all packets with a number less than or equal (modulo 65536) to the packet number in the receipt have been successfully received.

There is another receipt-like signal called an acknowledgment. It is used to implement flow control via a windowing mechanism now popularized through HDLC protocols. An acknowledgment obviates the need for sending receipts. Each higher-level protocol has a pre-adjusted window size, and although there is a facility for communicating dynamic adjustment of windows, this is never done in practice, to avoid introducing complexity into the protocols.

e) HIGHER-LEVEL PROTOCOLS

Higher-level protocols appear to the user as utility programs that can be invoked via a single command, or in the case of the lisp machines, by creating a session window of an appropriate type. These protocols include STATUS, MAIL, SEND, DOVER (for generating hard copy on the main laser printer), FINGER, TELNET (as in the Arpanet remote login protocol), Arpanet GATEWAY, and SUPDUP (which is the preferred remote login facility of the Chaosnet), and TIME. STATUS is used to generate a message which must be answered by every entity on the network and is used for maintenance purposes, MAIL is self-explanatory, SEND is used for interactive messages between users, FINGER provides information on a given user, Arpanet GATEWAY allows a Chaosnet user to access any Arpanet service through

one of the PDP-10s which act as Arpanet gateways, TIME provides the number of seconds elapsed since January 1, 1900 Greenwich Mean Time as a 32-bit number. Lisp machines, which do not have hardware calendar clocks, use TIME to find out the date and time when they first come up for service. The SUPDUP protocol is discussed later in this paper; it is a ment of TELNET.

f) FOREIGN PROTOCOLS IN Chaosnet

Any foreign protocol based on the idea of a full-duplex stream (or 2 half-duplex streams) of 8-bit bytes can be simply incorporated into Chaosnet using the Chaosnet connection mechanism instead of whatever stream protocol the foreign protocol was originally using. This was the case with Arpanet's TELNET.

There is a facility known as the Chaosnet foreign-protocol protocol which allows alien packets to be transmitted through Chaosnet or allows two Chaosnet hosts to speak in non-Chaos manner to each other. Occasionally non-stream I/O devices ,such as graphics tablets, need to be connected to the network and this facility makes it possible.

5. SUPDUP -- Chaosnet's remote login facility

The SUPDUP's main claim to fame is that it allows for remote logins in a manner of terminal-independent output. This way, only the local system needs to know how to handle the user's terminal. SUPDUP also has a built-in graphics interface as well as local assistance for the remote text editors (i.e., text editors used on the remote machine by the local user). SUPDUP means to be a superior ment of the Arpanet's TELNET.

Both TELNET and SUPDUP define a virtual terminal, but TELNET's is a simple teletype, whereas the SUPDUP defines a display terminal optionally capable of involved operations on text, pictures, and the user-transparent sharing of the work of a remote text editor.

The trade-off associated with SUPDUP is that the remote operating system must be able to talk to SUPDUP's virtual terminal. Alas, most operating systems understand just printing terminals, leaving all other tasks to user programs. Since the SUPDUP's virtual terminal is not a superset of the ordinary teletype, such operating systems cannot communicate with it. Instead, user programs must create a SUPDUP server. To put this in perspective, the SUPDUP protocol is meant as a ment for having to know exactly everything about all types of terminals connected to a system; the annoying symptom of this is the PRIMOS Emacs' prompt for supplying the terminal type the user is on, or the Termcap feature of Unix.

From the user's point of view, SUPDUP is wonderful. It "magically" provides the right translations so that a remote login user need not manually set switches every time a remote login to a different host

is made.

SUPDUP also defines a META key (present on Ann Arbor terminals as PAUSE and also available on lisp machines), which behaves exactly like the CONTROL key. Thus, Emacs users can perform in one keystroke (META-something, META-CONTROL-something) commands which in other circumstances would be prohibitively long, i.e., require a prefix or two.

SUPDUP also provides flow control for non-networked terminals in a manner superior to that of the XON/XOFF mechanism which robs the user of all editing commands using CONTROL-S and control-Q. It is akin to the flow control mechanism of TCP. It works by means of allocation, which is stored in the host, and which is the number of characters that can be accommodated by the terminal short of overloading it, thus it acts much like network data layer windowing.

The graphics protocol of SUPDUP has the nice property that the terminals using it are not aware of whether they are used remotely or locally. The graphics connection does not need any additional network connections, nor is there any need for preparation--graphics may be initiated any time during text transmission. The graphics protocol can work with either display list or bit-matrix terminals, allowing, for example, drawing pictures on part of the screen and keeping the rest of it for text display without mixing the two.

The local editing protocol of SUPDUP is meant to improve the response time in running a display-oriented text editor on a remote system via SUPDUP. It allows the local computer (or terminal) to perform most of the user's editing commands, with the source of the editing power moving between the local and remote system in a manner invisible to the user. In practice, this has great effect

on the user response time, especially when comparing the performance one is accustomed to when editing thorough a remote login TELNET interface.

Basically, the remote system's editor must explicitly request synchronization with the local editor via the SUPDUP protocol, and then grant permission to the local editor to do editing. When the local editor is done, it relays its results to the remote system for storage and updating the state of the editing session as seen by the remote editor. In practice these results are relayed every few seconds, which is meant to reduce overhead in timesharing systems by reducing process-switching (i.e., the remote editor runs for fewer longer sessions rather than for more shorter ones). There are provisions for discarding local editing if the remote editor has reason to intervene (e.g., emergency back-up) while the local editor is working.

There is also a line saving protocol in SUPDUP whose mission is to tell the local terminal to hang on to a copy of some displayed information and later refer to that copy for redisplaying. This mechanism saves on retransmission. This is especially noticable when using scrolling commands within a remote editor in a SUPDUP session.

Lastly, SUPDUP is smart enough to remember prior remote logins by the user who is still logged in at its local host, and it will not prompt for password again. While this may not be a good feature in a security-minded environment, it certainly makes life easier and faster when one has to repeatedly remote login through layers of remote hosts (a practice normally discouraged for efficiency reasons), or when repeatedly opening sessions to the same remote host.

6. User's impressions of the Chaosnet/SUPDUP MIT environment

There was a time when MIT AI Lab's computing environment was regarded as the best in the world. That assessment is no longer uttered by quite as many people these days, and I think the main problem is the fact that the computing environment is outgrowing its local area network.

Having file-transfer and remote login packets carried on the same cable may not be particularly burdensome to the remote login user if the number of file transfers is small. This is not the case at the joint MIT computer labs anymore. Quite often just a few minutes before 5 p.m., when many users routinely perform file transfers to save their lisp machine buffers on their file servers, the networked terminal user and the remote login user can go many agonizing seconds without having their keystrokes echo on their screens. This is a direct result of the simple policy that once a node has grabbed a net, it can have it for the duration of its packet, and packets can be quite large. A high density of such occurrences, given the very short time slot that defines an idle net, can virtually shut out the interactive user on some occasions.

As the number of machines attached to the Chaosnet and other local nets at the joint MIT computer science labs grows, these symptoms will only become more acute. Also, the sheer complexity of having to deal with many hosts requires a non-trivial familiarity with the (in)conventions of many operating systems. The absence of having a virtual file system is becoming more painful each year, especially to new users.

The robustness of Chaosnet, which is the flip-coin side of the simplicity of its design, is unquestionable. In my one year at the AI Lab I do not recall a single instance of the net being down. (Though, I do remember various bridges giving out from time to time, but no more or less often than "normal" PDP-11's.)

One also misses the ingenious behavior of SUPDUP once one grows accustomed to the conveniences it provides. Using things called "Emacs" under Unix (or VMS) and Sytek has a positively neolithic feel to it in comparison to Emacsing on Chaosnet under SUPDUP.

The remote login facility works very nicely, as does the file transfer protocol, provided the two do not go head-to-head in a rush hour setting. It is very convenient having the ability to send mail and messages, as well as files, in an unrestricted way while remotely logged in.

The Chaosnet's main tenet, that of providing the lisp machine user with an umbilical cord to its mother file server, is proving itself every day. It is only when the network is stretched past its original design goal to accommodate terminal traffic that the user has any reason to complain.

I am convinced that the bulk of what local networking does at MIT is done if not optimally then at least satisfactorily. Considering the huge number of interfaces to so many different operating systems and peripherals, it is a winning system. However, considerable changes will need to be made in the way interactive computing is handled across the network to keep the present day situation from deteriorating. Who knows, it might be even possible to improve it.

8. Other network projects at MIT

Currently, MIT at large is developing a personal workstation net consisting of IBM PCs, DEC VAXes, DEC Professional microcomputers, and the Ethernet network, which will provide service to the thousands of faculty and students of the Institute. This project is called Project Athena, and it will be gatewayed to the Chaosnet.

Already there are some objections over the choice of processors for the network. People with experience in local networking at Yale and Brown claim that their Apollo stations are the main limitations of their computing power. (The Apollo workstation is a conventional (non-LSI) implementation of the Motorola MC68020 specificiations, sold for $40k.) Their claim is that Project Athena is selling itself short on CPU cycles before even starting up.

My main worry with Athena, while I share the Yale-Brown opinion, is that it uses the same file transfer/interactive computing sharing of the net as Chaosnet does. No doubt the file transfers on Athena will be mitigated in size by the memory/disk size of its personal workstations (vs. Chaosnet's lisp machines), but the sheer number of them (thousands) will make up for the slack. I feel that any user unlucky enough to be logged in to a networked terminal under Athena (say, talking to one of its VAXes), would get lousy service.

Independently of the Project Athena, both of the MIT computer labs are putting in new, separate Ethernets meant to install in every LCS office a bit-mapped display to be used with a time-sharing CPU such as the MIT-XX, a DecSystem-20, or a Vax 750 under 4.2bsd Unix in single user mode, and likely a TI Explorer Lisp Machine, and at the AI Lab, a 3640.

I think this is the right way to go. This would enable putting the networked terminals only on a subnet, possibly improving response time for all, distributing the known bottlenecks over more cable. The trend to have people work all day on dedicated lisp machines of a Symbolics 3640 class brings us one more step out of the dark ages of interactive computing, the age of the hardwired time-shared terminal. And, with projects such as the Gerald S. Sussman's current attempted construction of a lisp machine suffused with RAM (eliminating all need for virtual memory management), and with the already manifested trend to have users save files on their local lisp machine's secondary storage (disk or whatnot), we are compelled to change our view of what the mission of networks is from that of helping make do with resource limitations to that of being mere passageways.

9. Bibliography

In order of utilization by this paper:

"Chaosnet", David A. Moon, AI Memo #628, Massachusetts Institute of Technology, June 1981.

"The SUPDUP Protocol", Richard M. Stallman, AI Memo #644, Massachusetts Institute of Technology, July 1983.

"A History of Networking at MIT LCS/AI Lab", an LCS seminar by Dave Clark, MIT Lab for CS staff member, Massachusetts Institute of Technology, summer 1984.

"The Lisp Machine Manual", David Weinreb and David Moon, internal publication, Massachusetts Institute of Technology, The AI Laboratory.

"The Athena Project: A Seminar Series and a Public Forum", invited lectures given by computer network experts (of particular interest: "Project Iris, Brown University"), Independent Activities Period, January 1984, EECS Department, Massachusetts Institute of Technology

"C690 Seminar: Computer Networks and Communications", Professor Frank F. Prosser, Fall Semester1984, Indiana University Computer Science Dept.

Personal Communication sources, in alphabetical order:

Alan Bawden, MIT AI Lab, September 1983 - August 1984.

Bernie Greenberg, Symbolics Inc., September 1983 - August 1984.

Scott A. Jones, MIT AI Lab, August 1984 - May 1985.

Christopher Lindblatt, ibid, August 1984 - May 1985.

David Plaisier, Indiana University CS Dept., June 1985

Jerry Roylance, MIT AI Lab, September 1983 - August 1984.

Steve Strassmann, ibid, September 1983 - December 1984.

Jonathan Taft, ibid, September 1983 - August 1984.

Gail Zacharias, ibid, December 1984 - January 1984.

The author wishes to thank Professor Marvin Minsky for the invitation to and generous support while at the MIT AI Lab, September 1983 to August 1984, and to Professor Douglas R. Hofstadter for making it possible. Th research conducted at The Massachusetts Institute of Technology's Artificial Intelligence Laboratory is supported by The Office of Naval Research, United States Department of Defense. The views expressed herein are those of the author alone and the references indicated above do not imply anyone's endorsement of these views. The

technical specifications and information cited in this paper is solely based on the references cited above.

I would also like to express my heartfelt thanks to Professor Frank F. Prosser for getting me interested in writing on the issues of networking, for encouraging me to write this paper, and for having had patiently proofread it.