ELECTRONIC SYSTEMS LABORATORY
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Memorandum MAC-M-394

CONNECTING M.I.T. COMPUTERS TO THE ARPA COMPUTER-TO-COMPUTER COMMUNICATION NETWORK

Dietrich Vedder
January 31, 1969

ABSTRACT

The Advanced Research Projects Agency of the Department of Defense is planning to connect computing facilities of different academic and industrial institutions by a nationwide network. MIT may connect either the GE 645/Multics System or the IBM 7094/CTSS System to this network. This report explores some of the network and routing problems of a computer-to-computer communication network, and outlines the necessary hardware connection for the GE 645 and for the IBM 7094. For the GE 645, it seems best to connect the IMP (Interface Message Processor) of the ARPA Network to the Direct Common Peripheral Adapter (High-Performance Channel) of the GIOC. For the IBM 7094, it is necessary to connect to Channel D, presently occupied by the PDP-7/Kludge. A small message-distributing computer is proposed to provide ports for an IMP, the PDP-7/Kludge, the PDP-9/Kludge, and up to 15 ARDS terminals.

Submitted as Project Report for 6.842

Supervisor: John E. Ward

Work reported herein was supported (in part) by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number NOnr-4102(01). Reproduction in whole or in part is permitted for any purpose of the United States Government.

I. INTRODUCTION

The Advanced Research Projects Agency (ARPA) of the Department of Defense is planning to connect computing facilities of different academic and industrial institutions by a nationwide network. This network is to be used as a research tool to study computer communication problems. The network should also make unique computing facilities present in only one center available to many computation centers.

The network is of the store-and-forward type, in that a message is sent from one nodal point of the network to the next and is stored there before it is sent on towards its destination. Each nodal point consists of a so-called Interface Message Processor (IMP). The IMP's in the network are connected by 50-kilobit data links that make up the branches of the network. In addition to the nodal IMP's, each computing facility (called a host) connected to the network will have an IMP associated with it, and this IMP will perform most of the reformatting and message handling for the host. Each host will connect to the 50-kilobit network only through its own IMP.

M.I.T. is to be part of the network just described. It is not clear at this time, however, whether the computer to be connected to the network will be the GE 645 (Multics), or the IBM 7094 (CTSS), or possibly both. Also, little is known about the ultimate configuration of the IMP and of the network outside of the general specifications given in the initial proposals. Despite these questions, it was felt desirable to investigate the hardware implications of connecting M.I.T. computers to the ARPA network.

Section II discusses some general network considerations, including the problem of system start-up, and also presents some ideas on network structuring to avoid the classic "shuttle" problem in message routing. Section III discusses possible IMP/645 interfaces, and concludes that the Direct Common Peripheral Adapter (High-Performance Channel) of the 645 GIOC should be used. Although no specific information is available on the IMP I/O system, a cost of $2,100 in parts is estimated for construction of an IMP/HPC interface.

Considerations in connecting an IMP to the 7094 are presented in Section IV. It is concluded that a small message distributing computer added to Channel D could handle an IMP, the two buffered displays (PDP-7/Kludge, and PDP-9/Kludge), and a number of additional ARDS display terminal parts.

II. THE NETWORK

A. General Network Considerations

Messages from one host to another are to be handled in the following manner. A host will send a message to its own IMP via a host-IMP interface. This message should not be longer than 8192 (2^{13}) bits.* The message is then reformatted in the host's IMP to be compatible with the communication network, and it is split into small packets of uniform size. Each packet is a small message that has a header including such information as the address of the destination host, the address of the sending host, and an identification tag designating it as, say, the fifth packet belonging to a certain message of a total of nine packets. All packets belonging to the message are then sent forward according to some priority scheme and will arrive at the IMP of the destination host (not necessarily in order, since different packets may have traveled different routes). The IMP at the destination host will re-assemble the different packets belonging together into a message, reformat the message if necessary to be compatible with the destination host, and send the message to the destination host.

It is clear that an IMP in the network has only one major function, since it is only connected to other IMP's, while an IMP connected to a host has two major functions:

1. It must handle the processing associated with sending messages between it and other IMP's. This function involves:
   a. Address of decoding and routing
   b. Storage of message packets in transit

---

*This is one constraint put on the system to prevent overloading.

c. Handling of priorities among the different messages in transit

d. Generation and processing of messages controlling message traffic between IMP's.

2. It must handle the processing associated with sending messages between it and its host. This function involves:

a. Reformatting (if necessary) of all messages leaving and entering the host

b. Assembling messages to the host from packets received from the network, and splitting messages from the host into packets to be sent out on the network

c. Control of communications between IMP and host.

How the individual functions are handled in each IMP is determined a lot by general network philosophy. The part of the report directly following therefore deals with some of the issues such as starting-up IMP's in the network, structuring the network, and routing algorithms applicable to the network.

B. The Start-Up Problem

Starting up a computer involves a definite number of steps. First, execution must be stopped at the beginning of the start-up sequence, and the computer must be initialized. Then a program is forced from some external source into a known part of memory; and the computer is then started at a specific point of that program. From then on the computer is under program control and in the run state. It can now bring other portions of a control program into memory, or it can execute programs; in other words, it can do what it was programmed to do.

The important point in this process is that the computer must be forced into a known state by external means. The simplest way to accomplish this if, of course, by having an operator do it manually. Alternatively, a second computer can perform the start up if it is directly connected to the

first computer. For example, it is relatively easy to let a host start its own IMP, since we can have a special interface with special signals to implement start up.

It is a little harder to load a network IMP (not connected directly to any host) from another IMP, since we now require that:

1. all IMP's have their power turned on (this certainly must be done manually),
2. the communications adapter of the distant IMP to be started must be functioning, i.e., it must be able to synchronize itself to a received string of data,
3. the communications adapter must have hardware to recognize a special character, which would cause the IMP to be initialized and stopped and which would set up the hardware in the communication adapter such that the subsequent characters in the start-up transmission are deposited in some known area of memory,
4. at the end of the transmission the IMP must be started at some know location in memory.

The special character for start up could be some control character preceded by DLE, similar to an STX (start of text) or an ETX (end of text) preceded by a DLE.

It should be apparent than at IMP not associated with a host can only be started directly by an adjacent IMP (unless it is done manually), since the start-up sequence cannot travel through another IMP without starting that IMP. If we want to make one host responsible for starting the entire network of IMP's, then that host can only do the job indirectly (unless each IMP has an individual hardware-recognizable start-up character) by telling some other IMP to start IMP's adjacent to it.

The program for an IMP not associated with a host should come from an adjacent IMP. Since the programs of adjacent IMP's are identical (or at least can be made identical except for some routing tables), there is no problem in starting one IMP by an adjacent IMP. For host-associated IMP's,

however, it does not make sense to get the program from an adjacent IMP. This is because at least the host-associated part of the program is probably unique and should be stored in the storage system of the host to which it belongs. For maintenance reasons alone, this program should reside in the host's storage system.

Now, should each host start its own IMP, or should one designated host start all IMP's, and then have each IMP associated with a host call on its host for the rest of its program. This author feels strongly that each host should start its own IMP. Some of the reasons why it should be this way are given below.

C. Some Thoughts on a Growing Computer Communications System

The research objectives for the ARPA network are stated in quite general terms. It is therefore not clear to the author whether these objectives are at all directed toward finding some generally applicable solutions for a large nationwide computer-to-computer communication network.

A large network in this case means a network of such a size that each IMP cannot easily keep in its memory all the information about how all other IMP's are interconnected.

Certainly the problems in creating such a network are numerous. Standards for message format and communications protocol* must be strictly adhered to.

The network configuration and the routing algorithms must be structured in such a manner that the system can grow easily; and it is this area of problems, which is expanded a little bit below.

Furthermore, it is not clear whether the store and forward system chosen is ultimately the best for the job. A switched channel system, which would provide a direct route for the duration of a transmission, using either analog carrier facilities or pulse code modulation carrier facilities are certainly feasible, but not as readily available at this time, as is a store and forward system.

---

*A.K. Bhushan, R.H. Stotz, "Message Format and Protocol for Inter-Computer Communication."

In the initial proposals for the network not much is said about network structure and routing algorithms outside of the statement that an interconnection table of all IMP's exists in each IMP, and that each IMP is to generate its own routing table. Some of the questions that come to mind are: as the system grows, does each IMP still store a complete interconnection table? How do we prevent a system failure due to two IMP's trying to deliver messages to each other, even though both IMP's decided they are too full to receive any more messages? How do we prevent two or more IMP's sending the same message back and forth, because other channels leading towards the addressed destination are busy?

The last question leads to the famous "shuttle" problem that arose in the toll network of the telephone system. This problem can be described by a simple example. Let us assume a party in Boston starts a telephone call to a party in Cleveland. The call is routed to New York, since Boston does not have any direct circuits to Cleveland. New York does have circuits to Cleveland, but they are all busy, so the decision is made to go to Philadelphia, since circuits exist from Philadelphia to Cleveland. It could turn out that Philadelphia in testing its circuits finds them all busy and returns the call to New York, since it knows also that circuits exist from New York to Cleveland. If this is allowed to happen, all the circuits between Philadelphia and New York could be made busy by one call that does not have a chance to be completed, and the traffic between Philadelphia and New York greatly disturbed. As will be seen, the telephone system has been structured so this does not happen.

The analogous situation may occur between two IMP's that are trying to send a message to a third IMP yet find the channels to the third IMP temporarily busy (the IMP's could be told of a permanent outage) and as a result shuttle the message back and forth, thereby possibly denying the channel between them to other messages that have a chance of getting through.

One way to solve this problem is to order all the nodal points in the network (i.e., central offices in the telephone network, IMP's in the

computer communication network) in a tree structure having several levels of hierarchy. Figure 1 shows such a tree structure which is completely analogous to the tree structure now existing in the telephone system in the USA and also in European countries. The endpoints in the tree structure are denoted by capital letters A through H. Each endpoint homes on so-called primary centers, which in turn home on sectional centers, which in turn home on regional centers. Every one of the regional centers (there may be more than two) is as a rule directly connected to all other regional centers. Also, it is possible to skip levels in the hierarchy, such as endpoint E and primary center e directly homing on regional center f.

The tree structure is used in routing algorithms by first defining a backbone route, which has the property that it goes up and down in the hierarchy only once. Short cuts are not allowed in defining a backbone route. For example, the backbone route from A to D is A-a-d-b-D; the backbone route from C to F is C-b-d-f-g-h-c-F and not C-b-h-c-F or C-F.

In selecting an actual route, it is allowed (and in fact preferred) to skip as many nodes or centers in the backbone route as possible; however, it is not allowed to add any new nodes not in the backbone route to the route selected, nor select any nodes in the backbone route out of sequence. Thus, the only allowed route from B to G is B-a-d-f-g-h-c-G. The route B-a-d-b-h-c-G is not allowed, even though it has one fewer node than the backbone route, since b is a newly added node. The shuttle problem discussed above is the reason why that rule is imposed on the network. Since center b is allowed to go to d on its way to G, center d must not be allowed to go to b on its way to G.

If C wants to send a message to F, it can do so via three routes. C-F is the first choice, C-b-h-c-F is the second choice, and C-b-d-f-g-h-c-F is the third choice and identical to the backbone route.

Note that as traffic warrants, nodes in the lower levels of the tree structure may be connected directly and thereby alleviate traffic bottlenecks in the upper levels of the structure. In fact any node in the

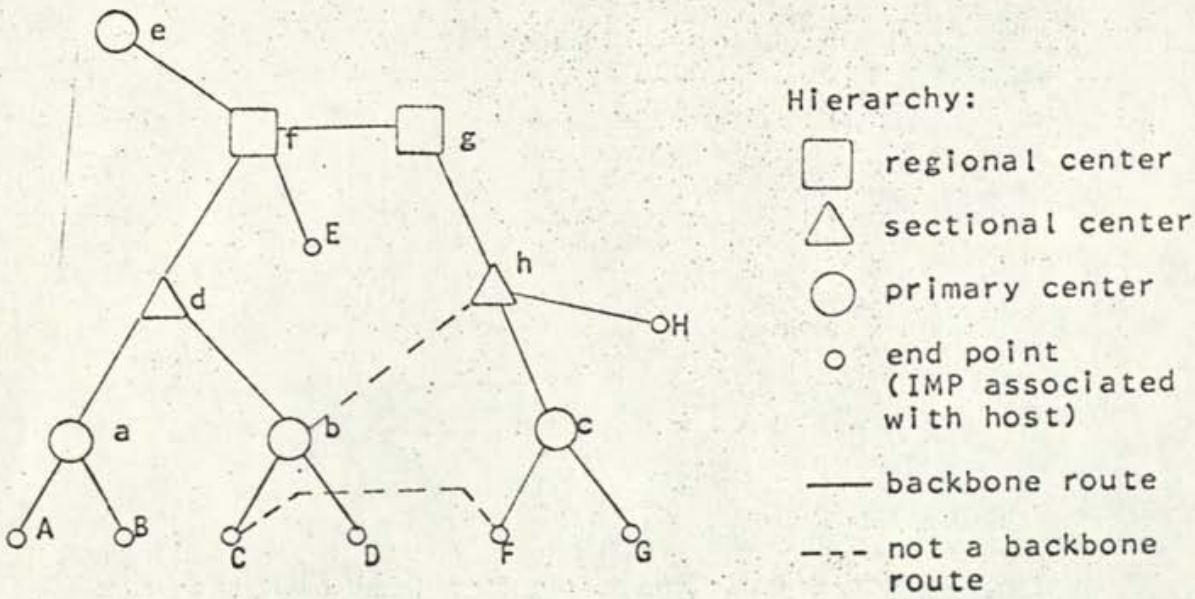

Figure 1 Hiearchical Tree Structure for Message Switching

network may still be connected to any other node in the network; the tree structure is able to accommodate all these links.

The tree structure can solve very nicely the problem of the nodal IMP's filling up with undelivered (or undeliverable) messages by use of the following rule. Given two IMP's that are connected to each other, the one having a higher status in the hierarchy can prevent the lower one from sending any messages to it, but the lower one cannot prevent the higher one from sending any messages to it. If two IMP's have equal status, such as centers f and g in Fig. 1, then one is picked arbitrarily as having a slightly higher status. This rule ensures that the higher levels of the hierarchy, where traffic tends to build up first, can get rid of their messages and thus stay operational.

It can happen, however, that specific bottlenecks develop because certain hosts or IMP's are out of service and are unable to receive messages. In this case, the hosts or IMP's that are unable to receive must be recognized, and any messages going to these computers must be turned back or rerouted. All the operational IMP's on the route of the unsuccessful message should then be informed that all messages with the same address as the unsuccessful message should either be rerouted or not accepted at all. In other words, some simple processing should be done on why a message cannot be delivered, and appropriate action should be taken all along the route of the trouble-encountering message. There are now two rules that govern message transmission:

1. Any IMP in the hierarchy must under normal condition accept messages from an IMP of higher status directly connected to it, but not vice versa.
2. Any message that encounters blocking due to an out-of-service condition of some IMP or host must cause all IMP's on its route (up to the node where the blocking occurred) to be set so that they will either reroute future messages that have the same address or not accept them.

It is of course the second rule that makes the first rule workable. The first rule implies that a lower IMP must be able to accept messages at a certain maximum rate given by the number and capacity of the connecting channels to higher IMP's. It is the second rule that guarantees that the lower IMP's will be able to get rid of messages at the same or a higher rate than they are asked to receive at.

Once the IMP network has the described tree structure superimposed on it, it is not necessary for each individual IMP to know how all other IMP's are interconnected. Each IMP would have a translation table, which maps all addresses into specific actions, these actions being the transmission of a message via a certain route or the rejection of the message due to a wrong address. The translation tables can still get large if the addresses are not properly chosen, but splitting the addresses into groups helps to alleviate the problem. We only have to look at a telephone number being split into area code + office code + number to see how the problem could be solved. footnote!

D. Autonomy of IMP's

If a communication system such as the IMP network grows to a relatively large size, it is difficult to control the network from one point. Furthermore, it is not wise for reliability reasons to have just one point control the network. One should instead make an individual IMP virtually autonomous. Unless one wants to shut down the system at night and start it in the morning, an IMP will only be started up after some failure. But a failure will usually require some maintenance work, performed by a knowledgeable person. It makes much more sense to let him start up the IMP after correcting the problem than to have a distant host direct the start up of the machine. Furthermore, it should not be too hard to construct the programs in such a way that the basic program is identical (except for some data areas) for each IMP that is not connected to a host. It should therefore be easy to get the program for each individual IMP from an adjacent IMP.

Since each IMP directly connected to a host has some special programs to interface it to the host, it should be the host that should store the backup copy of the IMP's program. Also, since the host's IMP appears as an I/O device to the host and is physically near the host, it does make sense to start the IMP from the host, although it is not absolutely necessary to do it that way.

III. THE CONNECTION BETWEEN THE GE-645 AND THE IMP

A. Comparison of the Word Synchronous Adapter and the High-Performance Channel

The IMP will appear to the GE-645 as just another I/O device, and must therefore be connected to the General Input Output Controller (GIOC) of the GE-645. Two different adapters were considered for connecting the IMP to the GIOC: the Word Synchronous Adapter, and the Direct Common Peripheral Adapter (also called High Performance Channel = HPC).*

The Word Synchronous Adapter (WSA-600) is an indirect adapter, which means that it does not store its own Data Control Word (DCW). The GIOC therefore has to access memory three times for each 36-bit word it transfers for the WSA-600. The first memory access is used to obtain the DCW from the proper mailbox, the second memory access is used for data transfer, and the third memory access is used to store the updated DCW in its mailbox.

The WSA-600 is full duplex, therefore, data transfer can occur simultaneously in both directions. Seventy-two bits of storage is provided for each direction of transfer, which splits into one word of buffering and one word of shift register. Character assembly and disassembly is done on a plug-selectable basis for six + parity, seven + parity, and four-out-of-eight code. The WSA-600 is capable of transmitting and receiving at a maximum rate of 240 kilobits/second. If seven + parity code is used

---

*Actually a third adapter, the Custom Direct Adapter (CDA), was also considered. This adapter performs even better than the HPC (it transfers 12-bit, 18-bit, or 36-bit words at a time) but it would probably be more expensive to rent, since it has more equipment than the HPC (7 rows of modules for the CDA versus 5 rows for the HPC). The real problem with this adapter is the fact that no hardware exists yet; and it is less likely to be put into production than the WSA-600.

between the GIOC and the IMP, then a rate of 30,000 characters per second can be sustained. The major disadvantage of the WSA-600 is that it presently exists only in specification form; the adapter itself has not been completely designed.

The High Performance Channel (HPC) is a direct adapter. It therefore stores the 72-bit DCW and needs only one memory cycle to do a data transfer. A 36-bit single word or a 72-bit double word may be transferred at a time. The HPC has a buffer register of 72 bits plus another 72-bit character assembly area. Six bits plus parity are transferred at a time to and from the connected I/O device. Data is transferred in only one direction at a time. The maximum data transfer rate is 400,000 characters per second, which is equal to 2.4 million bits per second.

B. Some of the Reasons Why the HPC Should be Selected

The HPC is, of course, an existing and working circuit. On the other hand, the decision to complete development of the WSA-600 rests with the General Electric Company and will depend probably on their market predictions for the WSA-600. While a 50-kilobit data-set interface is the only I/O device that promises to be a standard feature of the IMP, favoring connection to a WSA-600 that has the same interface, it can be argued that a nonstandard HPC interface for a small computer such as the IMP can be designed and built relatively easily here at MIT.

Beside the problems of procuring the actual interfaces, questions of speed and efficiency greatly favor the HPC. We surely do not want to run the interface between the IMP and the GIOC at less than 50-kilobits per second. This is because the entire IMP communication system runs at that bit rate, and providing less than that between host and IMP would surely create a bottleneck. At the rate of 50 kilobits per second, efficient data transfer through the GIOC is important in order that the other I/O operations are not slowed down. The HPC is six times more efficient than the WSA-600 (this efficiency of the HPC is due to the

storage of the DCW in the adapter and the buffering and transfer to the GIOC of a double word instead of a single word). The actual maximum data transfer rate of the HPC is 2.4 million bits per second and is ten times as large as the maximum transfer rate of the WSA-600. This relatively high transfer rate is nice to have, especially if the IMP connected to the GIOC terminates several 50-kilobit lines to nearby universities such as Harvard and Dartmouth that ultimately may end up having a high traffic rate with MIT.

In short, the IMP should be connected to the GIOC via the HPC for three main reasons:

1. An interface circuit between HPC and IMP can be procured relatively easily.
2. The GIOC operates much more efficiently if the IMP is connected via an HPC.
3. The maximum data transfer rate is high enough to avoid bottlenecks, even if traffic into and out of the GE-645 computer system is heavy.

It should also be said that estimates of monthly rental charges for the HPC and the WSA-600 are about equal; so there is no cost disadvantage in picking the HPC.

C. Interface Requirements for the HPC

The interface requirements are specified in detail in the General Electric Product Performance Specification for the "Common Peripheral Interface" (43A130524). Answers to any detailed questions should be found in this document. Only a general outline of the interface and of the hardware required to connect a small computer and the HPC will be given here.

The following leads interconnect to the HPC:

Lines from the HPC:

1. There are seven information lines from the HPC (6 data lines + parity).

2. Three lines from the HPC are used to control data transfer operations: the Read Clock Line, the Write Clock Line, and the End Data Transfer Line.
3. The I/O Line from the HPC is used to send commands from the GE-645 to the peripheral (in this case the IMP).
4. A Program Load Line from the HPC is used to request the peripheral to send one record for bootstrap loading and similar purposes. This line is probably not necessary in the IMP interface.
5. A peripheral Reset Line from the HPC tells the IMP that the GE-645 is operating or not operating.

Lines to the HPC:

1. There are seven information lines to the HPC (6 data lines + parity).
2. Four Major Status Lines transmit status information to the HPC.
3. Three lines to the HPC are used to control data transfer operations: the Read Clock Line, the Write Clock Line, and the Terminate Line.
4. A Special Interrupt Line to the HPC allows the IMP to demand action from the GE-645.
5. An External Reset Line to the HPC tells the HPC whether the IMP is operational or not.

All lines between the HPC and the IMP (except the Major Status Lines to the HPC and the External Reset Lines in both directions) carry pulses and must conform to the specifications set by General Electric. On the IMP side, all of these lines must therefore have the proper pulse drivers or pulse receivers as appropriate. The Major Status Lines carry levels instead of pulses and must be equipped accordingly.

The External Reset Leads in both directions have a relay signaling system. An Enabled condition is signaled if the center conductor of the External Reset Lead is connected to the shield, and a Disabled condition

is signaled if the center conductor is not connected to the shield. The connecting function is done by mercury wetted contacts ensuring clean switching.

The External Reset Lead keeps its respective side in the reset state as long as the other side has not signaled the Enable condition. The External Reset Lead thus accomplishes the suppression of line transients while the HPC and the IMP are disconnected or while one side has its power off or is disabled for some other reason. Note that the signaling convention very nicely takes care of the disconnect or power off condition, since the External Reset Lead will be open and thus will keep its respective side reset.

The Communications between IMP and HPC can be implemented by using commands from HPC to the IMP and status information and interrupts from IMP to HPC.

The HPC (and the GE-645 behind it) is, of course, in control of the communications between HPC and IMP; it can send commands to the IMP interface initiating appropriate operations. The IMP interface can send status information back to the HPC, thereby causing action indirectly. Similar to teletype consoles, the IMP would also be able to send an interrupt to the HPC and thus initiate new action.

Commands are sent from the HPC to the IMP on the seven information leads. Status is sent back to the HPC via the four Major Status Lines and also via the seven information leads. The detailed procedure for passing commands and status information is given in the previously mentioned General Electric document. Also, some restrictions and conventions in assigning meaning to the command and status words are given in that document.

D. An Estimate of the Cost of an HPC/IMP Interface

One cannot design a specific interface without knowing the detailed specifications of the IMP; but an estimate of the cost of an IMP interface is possible, since most small computers have similar I/O features. It is assumed in this estimate that the IMP has a feature similar to the

Direct Memory Access feature of the PDP-9. (The 8-million bit maximum transfer rate specified for the IMP-host interface virtually dictates such a feature.)

The following is a list of essential hardware for the interface:

7-bit register	(Receive and transmit buffer, pulse receivers included)
12-bit register	(Two-character buffer)
4-bit register	(Major Status)
7 pulse drivers	(7 information lines to HPC)
4 cable drivers	(to transmit Major Status)
2 mercury relays	(for External Reset Lines)
4 pulse receivers	(for pulse receiving on control leads)
4 pulse drivers	(for pulse sending on control leads)
3 flip-flops and 18 gates	(to distribute character over entire word)
13-bit register	(address register for PDP-9 DMA)
control logic	

From the above list we can estimate the cost assuming DEC modules.

Items	Cost for Each	Total Cost
39 flip-flops	$15	$ 585
11 pulse drivers	11	121
2 mercury contact relays	10	20
18 gates	5	90
4 cable drivers	12	48
4 pulse receivers	15	60
control logic	--	500
2 mounting trays	142	284
I/O bus	--	380
		$2,088

Thus the cost of an IMP interface is about $2,100 not counting any wiring, installation, and engineering cost.

IV. THE CONNECTION BETWEEN THE IBM-7094 AND THE IMP

A. General Discussion

The IBM-7094 with its time-sharing system CTSS is an established system with a number of capabilities that may prove useful to the community of users to be connected by the IMP network. It is therefore possible that the IMP network may at first be connected to the IBM 7094.

A hardware connection to the IBM-7094 is not as straightforward as to the GE-645. The only available high-data-rate connection to the IBM-7094 is channel D, and that channel is already used to handle the two existing ESL refreshed display consoles (Kludges). Figure 2 shows the present configuration, which at this time is completed except for the installation of the data link. In this configuration the PDP-7 is used to handle the Kludge 1 and is also used to transfer messages and display lists from the 7094 to the PDP-9 and vice versa. If we want to connect the IMP, we have to use channel D, not only for the IMP, but also for Kludge 1 and Kludge 2. Furthermore, we possibly would like to connect a number of ARDS ports to channel D, since the number of ARDS ports provided by the IBM-7750 is limited to at most eight ports (four at the moment).

There are at least two possible ways to handle this channel multiplexing problem. One can use the PDP-7 as the multiplexing computer in addition to its task of refreshing the Kludge 1, or one can use a separate message distributing computer. Technical considerations favor the separate message distributing computer slightly, since it represents a "cleaner" solution in that the different functions, such as driving a Kludge and distributing messages, are separated and since it is easier to troubleshoot and maintain the system.

It turns out that the economics also favor the separate message distributing computer. A look at the memory requirements makes this clear. If one wants to run up to 15 ARDS ports, about 3.3K words of storage are needed for the ARDS ports alone (See Appendix A). An additional 2K words of storage are needed for temporarily storing display lists or messages

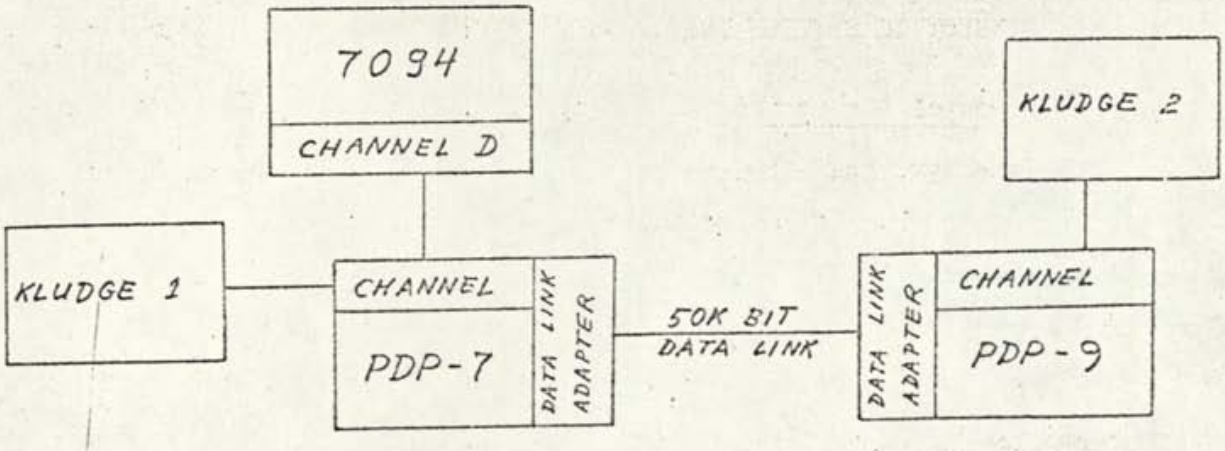

Figure 2 Present Equipment on 7094 Channel D

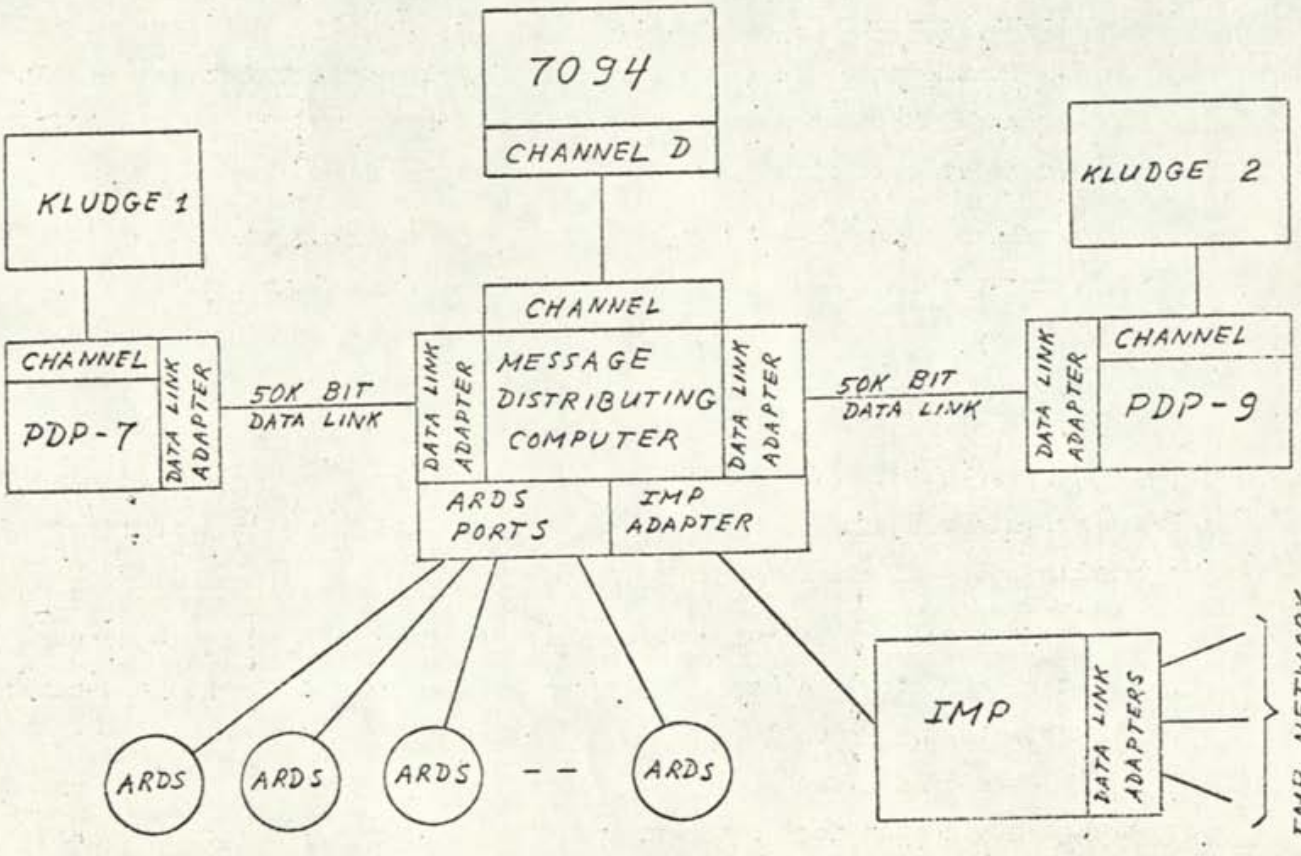

Figure 3 Proposed Message Distributing Computer for 7094

to and from the IMP. If the PDP-7 is to be the multiplexing computer, another 2K words is needed for the display list of Kludge 1. Since the present storage size of the BSL PDP-7 is 8K words, this would leave only 700 words for all programs in the PDP-7. An increase of the PDP-7 memory is therefore needed for this solution. Furthermore, it is not advisable to run the multiplexing program in the PDP-7 without a memory protect feature, since we want to allow Kludge users to modify programs in the PDP-7. As a result, if the PDP-7 is used as a multiplexing computer, we need at least another 4K of memory, a memory extension control, and a memory protect option. The cost of these features are as follows:

4K memory bank (installed)	$21,020
Memory extension control (installed)	8,340
Memory protect option (installed)	900
	$30,280

But for thirty thousand dollars we should be able to get a message distributing computer with the necessary I/O equipment to do the job. Figure 3 gives the configuration for a separate message distributing computer.

B. I/O Configuration of the Message Distributing Computer

Since both the PDP-7 and the PDP-9 already have 50-kilobit data link adapters, it is easy to connect them to the message distributing computer via 50-kilobit data links and data link adapters. Thus both Kludges can be moved around on the campus and do not have to be adjacent to the 7094.

Since the 50K bit data link arrangements are character oriented (7 bits + parity), there is a 160 microsecond time interval for responding to a data transfer request from the data link adapters, provided double buffering is used for the characters in the data link adapters.

The connection to the 7094 should probably be done via a channel very similar in structure to the channel built for the PDP-7. A direct

memory access feature in the small computer is very desirable for this purpose.

The ARDS ports should consist of half-duplex circuitry capable of receiving at 150 bits/second and transmitting at 1200 bits/second. Double buffering of characters in each ARDS port is not necessary if it can normally be guaranteed that the message distributing computer can respond to a data transfer request by an ARDS port within 800 microseconds. If this is not possible, then double buffering of characters will lengthen the required response time to about 7 milliseconds.

It is hard to say what the adapter to the IMP should look like, since a number of policy decisions are involved. If it is not considered essential that the IMP can be started remotely from the 7094, then a 50K bit adapter, even if used without a data link, is probably best, since both the IMP and the message distributing computer can easily be provided with an additional data link adapter. If remote starting capability is required, then the IMP interface must either be a channel interface that is capable of sending a special start command or a 50K bit data link adapter capable of responding to a special start character. It should be clear that the message distributing computer must also have the capability of being started remotely, if the IMP is to have that capability.

C. Features of the Message Distributing Computer

The message distributing computer should have a memory size of 8K to accommodate up to about 15 ARDS ports and one display list or IMP message simultaneously. The protocol for message distribution and memory allocation should be arranged to give messages from the IMP to the 7094 top priority, since we must ensure that the IMP network can get rid of its message (see discussion on IMP network and congestion).

An index register would be helpful to do the large amount of list processing. Efficient shifting operations are also necessary to do the conversion from characters to words and vice versa.

APPENDIX A

Required Memory for Operating a Number of ARDS Ports

If we want to connect a number of ARDS ports to the IBM 7094/CTSS through a message distributing computer, then the required memory in the message distributing computer depends a lot on how often we are willing to allow user programs in the 7094 to be swapped in and out of core. Transmission to ARDS is at a rate of 1200 bits per second. Since each character is 10 bits, this is 120 characters per second. If we can guarantee that normally 600 characters can be stored in the message distributing computer for any particular ARDS port (but not all ARDS ports simultaneously), then the time interval between output imposed swaps is normally longer than five seconds; if we allow ten seconds worth or storage, then this interval is longer than ten seconds, and so on.

The following assumptions were made in order to calculate the memory requirements for ARDS output:

1. The output bit rate is 1200 bits/second.
2. It is assumed that on the average only 20 percent of the ARDS terminals are outputting at one time.
3. It is assumed that the number of ARDS ports is 5, 10, or 15.
4. It is assumed for the purpose of calculation that the maximum number of characters stored per ARDS port is either 600 or 1200 (5 or 10 seconds worth respectively).
5. The behavior at an individual ARDS port is statistically independent from all other ARDS ports.

If we are willing to store a maximum number of 600 characters (6000 bits) for an individual port, then the average number of bits per port is: $6000 \times 0.2 = 1200$ bits/port.

The standard deviation can be computed as follows:

$$\sigma^2 = E(x^2) - [E(x)]^2 = 6000^2 \times 0.2 - 1200^2$$

$$\sigma^2 = 5.76 \times 10^6$$

$$\sigma = 2400 \text{ bits (for one port)}$$

$$\sigma = \sqrt{n} \times 2400 \text{ bits (for n ports)}$$

If we are willing to store a maximum number of 1200 characters for an individual port, then the average number of bits is 2400 bits/port and $\sigma = \sqrt{n} \times 4800$ bits for n ports.

Below are two tables for a maximum storage of 600 and 1200 characters per ARDS port respectively, giving the required storage for 5, 10, and 15 ports.

	5 ports	10 ports	15 ports
max. number of bits	30,000	60,000	90,000
3 σ limit (bits)	22,000	34,800	45,900
3 σ limit (words) (14 bits/word)	1,570	2,490	3,290

Table 1. Maximum Allowed Storage per ARDS Port is 600 Characters

	5 ports	10 ports	15 ports
max. number of bits	60,000	120,000	180,000
3 σ limit (bits)	44,000	73,600	91,800
3 σ limit (words) (14 bits/word)	3,140	4,980	6,580

Table 2. Maximum Allowed Storage per ARDS Port is 1200 Characters

The tables show the absolute maximum amount of storage ever needed, and also the amount of required storage that we get if we add 3 standard deviations to the average amount of storage needed. The actual amount of storage needed at any moment is very unlikely to be above the

amount given by the 3 σ limit, provided the initial assumptions hold.

Note that the assumption on maximum number of characters per ARDS terminal was made only for purpose of calculating some figures on required memory. Now having the figures, we can turn around and say, given the amount of storage and number of ports in the table, it is very unlikely that we will be running into output storage limitations if a user program happens to generate up to but not more than 600 or 1200 characters of output at one time.

In the main report, 3.3K words was taken as a bogey figure for the amount of required memory for ARDS ports. This figure is based on 15 ARDS ports and maximum allowed storage of 600 characters per port (see Table 1). Note however that another 2K was reserved for display list storage. It so happens that display lists of this size are sent very infrequently; furthermore, we do not expect a large amount of IMP traffic initially. Therefore, an extra 2K of memory should usually be available for ARDS traffic, raising the maximum storage allowance per port to about 1000 characters per port. The initial 8K memory size requirement for the message distributing computer should therefore yield good response. In the future, when the system is running and IMP traffic increases to significant levels, actual traffic measurements can supersede the estimates above, and possible memory extensions can be planned more accurately.

A word about buffering inputs into ARDS ports is in order. Presently the IBM 7750 sends every full character received from a port on to the 7094 as soon as it is assembled. The storage requirements for ARDS input are therefore very small, unless we significantly change the present way or operating.