BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

Report No. 2918

NETWORK DESIGN ISSUES

November 1974

This research was supported by the Advanced Research Projects Agency of the Department of Defense and monitored by the Range Measurements Laboratory under Contract No. F08606-73-C-0027.

CAMBRIDGE WASHINGTON, D.C. CHICAGO HOUSTON LOS ANGELES SAN FRANCISCO

TABLE OF CONTENTS

page

Introduction

Over the past few years there has been much examination of the ARPA Network algorithms in view of operational problems observed, alternative algorithms chosen for other networks, and the large body of theoretical work on packet-switching networks being carried on at universities and elsewhere.

In this report we consider a few (by no means all) of the packet-switching design issues now under widespread consideration.

1. Design Requirements

We begin by giving the properties central to the ARPA Network design. The key assumption here is that the *packet processing algorithms* (acknowledgment/retransmission strategies used to control transmission over noisy circuits, routing, etc.) result in a virtual network path with the following characteristics:

a. finite, fluctuating, delay—a property of the circuits
b. finite, fluctuating, bandwidth—a property of the circuits
c. finite error rate (duplicate or lost packets)—a property of the acknowledgment (“ack”) system (this is a different use of the term “error rate” than in traditional telephony)
d. disordering of packets—a property of the ack system and routing

Duplicate packets are caused when an IMP goes down after receiving a packet and forwarding it without having sent the ack. The previous IMP then generates a duplicate with its retransmission of the packet. Packets are lost when an IMP goes down after receiving a packet and acknowledging it before the successful transmission of the packet to the next IMP. These four properties describe what we term the *store-and-forward subnetwork.*

There are also two basic characteristics of the source and destination IMPs in the virtual path described above:

e. finite storage—a property of the IMPs
f. differing source and destination bandwidths—largely a property of the Hosts.

A slightly different treatment of this subject can be found in [1] by L. Pouzin.

The fundamental requirements for the ARPA Network *message processing algorithms* are dictated by the six factors enumerated above. These requirements include:

a. Buffering—Because of the *finite delay* of the network, it may be desirable to have buffering for multiple messages in flight to increase throughput, that is, a system without buffering (one message at a time) may have unacceptably low throughput due to long round-trip delays.

b. Pipelining—The *finite bandwidth* of the network may necessitate the pipelining of messages through the network, by breaking them up into packets, in order to decrease delay. The bandwidth of the circuits may be low enough so that forwarding the entire message at each hop in the path results in excessive delay. By packetizing the message, the IMPs are able to forward the first packets of the message through the network ahead of the later ones. For a message of P packets and a path of H hops, the delay is proportional to $P + H - 1$ instead of $P * H$.

c. Error Control—The destination may need to exercise some controls to detect missing and duplicated packets, which would appear as incorrect data to the end user. Further, reports on delivery or non-delivery of messages may be useful, possibly to trigger retransmission. This mechanism in turn requires error control and retransmission itself, since the delivery reports can be lost or duplicated. The usual technique is to assign some unique number to identify each data unit and to time out unanswered units.

d. Sequencing—Since *packet sequences are received out of order* the destination must use a sequence number technique of some form to deliver messages in correct order, and packets in order within messages, despite any scrambling effect that takes place while several messages are in transit.

e. Storage allocation—The *finite storage* in both the source and destination means that each must exercise control over its use. The two basic approaches are for the source to hold a copy of the message which then allows the destination to treat each message as discardable if it does not have storage for it, and for the source to request or otherwise obtain reserved storage at the destination which removes the need for the source to hold a copy. The first approach allows low delay since there is no setup time in obtaining a storage reservation in the destination, while the second allows high throughput (if reservations are long-lived) by eliminating retransmissions caused when storage is over-subscribed. Of course, storage need not be committed at either the source or the destination if reliable transmission of messages is unimportant, and storage can be reserved at both source and destination for simultaneous optimization of delay, throughput, and reliability, although at some added cost.

f. Flow Control—The *different source and destination data rates* may necessitate implicit or explicit flow control rules to prevent the network from becoming congested when the destination is slower than the source. These rules can be tied to the sequencing mechanism (no more messages after a certain number are acceptable) or to the storage allocation technique (no more messages can flow until a certain amount of storage is free) or can be independent of these features.

In performing these functions, the message processing system is often exercising contention resolution rules to allocate scarce resources among several users. The twin problems of any such facility are:

- fairness—resources should be used by all users fairly.
- deadlock prevention—resources must be allocated so as to avoid deadlocks.

(We have also come to believe that it is essential to have a reset mechanism to prevent "impossible" deadlocks and other conditions that may result from hardware or software failures.)

Having considered the design requirements, we now turn to a number of specific design issues.

2. No Message Processing in the Subnetwork

There has been considerable discussion in the packet-switching community about the amount and kind of message processing that should be done in communications subnetworks. An important part of the ARPA Network design which has become controversial is the ARPA Network system of messages and packets within the subnetwork, ordering of messages, guaranteed message delivery, and so on. In particular, the idea has been put forth that many of these functions should be moved from subnetwork to Host level. As we attempt to show in this section, we feel the ARPA Network design is a good one, as good overall as any other recent proposal for a network design. We recognize deficiencies in the present implementation, but we feel that each problem is relatively easy to solve in a natural way within the framework of the philosophy of the present design. We also recognize that there are no absolutes at this stage in the development of computer networks, and we look forward with anticipation to the successful conclusion of the construction of those networks which have chosen divergent design principles so that we may better compare the results of differing design philosophies. Differing views on specific design choices in this area have been proposed by the designers of the Cyclades network [2], writers on the subject of inter-network communication [3], and ARPA Network researchers commenting on their experience [4], as well as others.

We summarize the principles usually given for eliminating message processing from the communications subnetwork:

a. For complete reliability, Hosts must do the same jobs and therefore, the IMPs should not.

b. Host/Host performance may be degraded by the IMPs doing these jobs.

c. Network interconnection may be impeded by the IMPs doing message processing.

d. Lockups can happen in subnetwork message processing.

e. The IMP would become simpler and have more buffering capacity if it did not have to do message processing.

The last point is true (if we can ignore the issue of whether the IMP *must* do message processing for its own control, statistics, and debugging messages), but the other statements are certainly subject to some question.

Much of the criticism of the ARPA Network design is based on misconceptions about the design or is criticism of specific network characteristics which could be fixed easily. For instance, a major motivation for many of the changes from the ARPA Network design has been the *perception* that the ARPA Network design limits bandwidth and suffers from lockups. This perception, however, is inaccurate; except for implementation bugs and deficiencies, the ARPA Network design is sound and free from lockups with no inherent bandwidth limitation—we have chosen the structure and operating parameters of the ARPA Network algorithms so that the limit they impose on Host throughput is higher than that imposed by the basic bit rates of the network circuits and Host interfaces. Another motivation has been the apparent coupling of the Hosts on an IMP in such a way that they can relatively easily interfere with each other because they share the same message number pipe to a common destination. In fact, such coupling is not fundamental to the ARPA Network design and there are plans to decouple the Hosts on an IMP. These two subjects are treated in detail in a later section.

Whatever is done at the Host level, it is critical for the communications subnetwork to handle its own problems or else it will perform poorly in many dimensions; e.g., message loss, message ordering, etc. Without such controls, the Hosts will be in conflict with each other, degrading their performance, with no way to resolve these conflicts. (For example, if the subnetwork handles congestion by merely discarding packets whenever congestion occurs, it is as likely to discard packets of “well behaved” Hosts, whose I/O rates are closely coordinated, as to discard packets from streams between Hosts which are not coordinating their rates.) The following reasoning indicates why this is so:

- The function of flow control is to prevent a source from overflowing the finite buffer capacity of a slower destination.
- All messages go through the destination IMP before going to the destination Host.
- Even with great buffering capacity, the IMP buffers would *sometimes* overflow.
- The Hosts are intrinsically unable to allocate IMP storage precisely.
- When destination IMP buffer overflow occurs, performance is drastically reduced because the source must retransmit, increasing delay and decreasing throughput.

In general, in large networks with many Hosts, the majority of Hosts will find a communications subnetwork insufferable if it is too prone to message loss. Despite the fact that some Hosts may not care how prone to message loss the subnetwork is, and despite the fact that many Hosts may need to duplicate functions also performed in the subnetwork for those (hopefully rare) instances when the subnetwork does fail, the majority of Hosts will want the subnetwork to do the best job it can so as to minimize Host performance of these functions. Every channel is noisy; and the IMP must do what it can to minimize noise. Furthermore, this is basically inexpensive to do in the IMP, especially when compared to the cost of doing it in *all* the Hosts and, generally, what is done at IMP level is in no way detrimental to the Host level, even if it does not help the Host level.

Moving these functions to Host level also does not solve the perceived efficiency and deadlock problems associated with performing these functions in the subnetwork. The problems, if they really exist, are merely moved out one level, although with larger memories in some Hosts, the problems may be less frequent. (In the ARPA Network case, the problems are also pushed to the fake Hosts [processes within IMPs] and the Terminal IMP.)

In the few areas where the present ARPA Network design does present real limitations to some desired types of network use (e.g., transmission of fixed, very low delay traffic such as speech), mechanisms can be added to the ARPA Network in parallel to the existing mechanisms for the purpose of accommodating these desired but presently unobtainable types of traffic service.

One can easily imagine that if very many of these functions were removed from the IMP and given to the Hosts a large number of Hosts would add a mini-Host between the IMP and Host to perform these functions. In fact, the trend today, with declining computer costs, is to remove functions from large computers and to put them in peripheral processors, not to add functions to the large computers. Notice, however, that if a peripheral processor is to take on the message processing jobs which the IMPs now perform, then it will require a large storage device to buffer out-of-order packets or else it will suffer the same type of contention for buffer space which occasionally occurs in the IMPs. In order to obtain desirable delay and bandwidth characteristics, such a peripheral processor will be obliged to work with the set of all other such processors and Hosts and reinvent mechanisms similar to those currently in the IMPs. But with many *independent* implementations of this kind, not only will initial implementation cost be high, but coordination and testing of any necessary or desirable changes will be extraordinarily difficult. This is in sharp contrast to the relative ease of making corrections, improvements, and even fundamental design changes in the subnetwork.

We now summarize our main contentions about the place of message processing facilities in networks:

a. A layering of functions, a hierarchy of control, is essential in a complex network environment:

   - For efficiency, IMPs must control subnetwork resources, and Hosts must control Host resources.

   - For reliability, the basic subnetwork environment must be under the effective control of the IMP program—Hosts should not be able to affect the usefulness of the network to other Hosts.

   - For maintainability, the fundamental message processing program should be IMP software, which can be changed under central control and much more simply than all Host programs.

   - For debugging, a hierarchy of procedures is essential, since otherwise the solution of any network difficulty will require investigating *all* programs (including Host programs) for possible involvement in the trouble.

b. The nature of the problem of message processing does not change if it is moved out of the network and into the Hosts; the Hosts would then have this very difficult job even if they do not want it.

c. Moving this task into the Hosts does not alleviate any network problems such as congestion, Host interference, or suboptimal performance but, in fact, makes them worse since the Hosts cannot control the use of IMP resources such as buffering, CPU bandwidth, and line bandwidth.

d. It is cheap to do message processing in the IMPs, and it has very few detrimental effects.

One party who has been particularly active in support of removing message processing altogether from the communications subnetwork has been Cerf, especially in his paper assessing the ARPA Network protocols [4]. Specific conclusions that Cerf reached in his paper include:

a. Multi-packet messages can be eliminated without loss of potential throughput;

b. Ordering at the destination IMP can lead to lockups, and since the Hosts must do ordering anyway if they are to recover in the cases where the subnetwork fails, the IMPs need not do it;

c. If the destination IMP doesn't reorder or reassemble, arbitrarily long messages are possible;

d. Retransmission, end-to-end acknowledgments, error detection, duplicate detection, and message ordering at Host level eliminate this need in the IMP subnetwork.

It is our contention that all four of these conclusions are flawed. At the risk of some duplication of what was said above, we respond to each of Cerf's points:

a. With regard to eliminating multi-packet messages while still maintaining high Host throughput, Cerf's conclusion is based on three ideas, to which we offer rebuttals:

    1) that use of multi-packet messages presents a buffering limit—the limit Cerf is concerned with is actually a bit-coding limit and one that can easily be removed;

    2) that Hosts use the network suboptimally since they have agreed among themselves to have only one message at a time outstanding (on a given "connection"); by using the network better, the Hosts could buy back any loss caused by eliminating multi-packet messages—in fact the Hosts could improve their use of the network, but this improvement should be added to the network's already good performance rather than wasted in recouping the loss in network performance resulting from removing multi-packet messages (the fact that they have not done so is at least partly due to the

difficulties in coordinating desirable changes to independent implementations which we discussed above) and,

3) that eliminating multi-packet messages would cause only minimal performance degradation—in fact, even without multi-packet messages the subnetwork would have to perform similar functions on groups of packets or else great loss in performance would be incurred.

b. With regard to the idea that ordering in the subnetwork can cause lockups:

1) the fundamental idea of the IMP reordering scheme wherein all destination storage is allocated is sound, and the notion that the IMP can lock up with out-of-order messages is incorrect;

2) doing reordering at the Host level is also prone to lockups. Although there is freedom at the outermost level to discard troublesome packets to resolve deadlocks, this will result in the same inefficiencies as it would if done too frequently at inner levels; and,

3) customers invariably want their packets and messages delivered in the order sent (potential commercial networks and military networks confirm this desire) and therefore this job should be done in a centralized place, for reasons of economy in implementation.

c. The idea of obtaining arbitrarily long messages is so weak that even Cerf notes its fatal flaw: it requires the nodes to know the Host/Host protocol(s). Furthermore, it is possible to have arbitrarily long data streams in the ARPA Network made up of finite-size messages.

d. The fact that the Hosts can do the same functions that are done in the subnetwork does not mean that they all want to, that it is as efficient for them to do so, or that even if they did do these functions the subnetwork would not still have to do them if it was not to perform very badly. In fact, it is clear that the subnetwork must protect itself against local congestion, mismatched input and output rates, and the like.

3. Single-Packet Messages Only

The following question is often asked: “If one increases packet size, and decreases message size until they become the same, will not the difficult reassembly and flow control problems be removed?” The answer is that, unfortunately perhaps, message size and packet size are almost unrelated to flow control and reassembly, and we discuss each of these four issues below.

Packet Size. Network delay is a complicated function of line speeds, propagation delay, utilization of the lines, the priority scheme used, IMP processing time, and the possibility of preemption. In the present ARPA Network, delay for a small priority message is, to first order, proportional to the packet size of the other traffic in the network. Thus, small packets are desirable. This attitude changes when lines become so long or fast that propagation delay is larger than transmission time.

Message Size. Message size needs to be large because the overhead on messages is significant. It is inefficient for Hosts to have too many message interrupts, and for the IMPs to have to address too many messages. The upper limit on message size in the current network is what can conveniently be reassembled, given IMP storage and network delays.

Reassembly. When a channel has an appreciable delay, several pieces of data must be in progress in the channel at one time in order to achieve full utilization of the channel. It makes little difference whether these pieces are called packets which must be reassembled or messages which must be delivered in order. In the ARPA Network without satellite links, the amount of data which must be “in the pipe” between the source and destination IMP to fully utilize it is on the order of thirty packets. With satellite links, many more packets are necessary. Unless one is willing to endure a three percent (100%/30) utilization of the source-to-destination pipe, there must be reassembly, in the IMPs, or if not in the IMPs, in the Hosts.

Flow Control. Flow control requires there to be space to hold all the traffic in the “pipe” should the destination suddenly slow its rate of acceptance of the traffic. Since there are many “pipes” in a network, flow control becomes complex. In the ARPA Network, flow control is currently done on “chunks” of that space which happen to be the same size as a message because that is a convenient size and simplifies bookkeeping. Of course, this is not entirely coincidental as the same forces work to determine reasonable sizes for both the message units and the units of flow control. Packet size seems to have little relation to flow control in the ARPA Network implementation.

4. Packet Size

There has been much thought given in the packet-switching community to the proper size for packets.

For instance, in his thesis [5], Metcalfe points out that larger packets have a harder time getting across an error-prone telephone circuit, and this drives the packet size down. At the same time, Metcalfe points out, overhead considerations (longer packets have a lower percentage overhead) drive packet size up. For a set of assumptions about the ARPA Network circuit error characteristics, Metcalfe then calculates the packet size which optimizes effective throughput (i.e., not too much overhead, and not too much lost to retransmissions due to errors). In fact, for his assumed error rates, Metcalfe concludes that the ARPA Network packet size of about 1000 bits is within the flat range on the optimization curve and is thus sufficiently well chosen.

At another point in his thesis, Metcalfe notes that store-and-forward delay is reduced as packet size becomes smaller. In other words, the delay-lowering effects of pipe-lining become more pronounced as packet size decreases. However, again, as the packet size goes down, effective throughput also goes down due to overhead considerations. In this case, unfortunately, we are trading off delay against effective throughput and it is not possible to define a natural optimum.

In their book, Davies and Barber [6] argue for a certain minimum length “packet” (about 2000 bits) because they have concluded that most of the messages currently exchanged within banks and within airlines will nicely fit in one packet of this size. This does not mean, however, that they believe ARPA Network packets are too small, since they use the term “packet” for the unit of information which is called a “message” in ARPA Network terminology, i.e., the basic unit of information passed from Host to IMP. In fact, an ARPA Network message clearly exceeds this recommended minimum size.

A recent paper on measurements by Kleinrock and Naylor [7] suggested that the ARPA Network packet size was suboptimal and should perhaps be reduced to about 250 bits. This was based on optimization of IMP buffer utilization for the observed traffic mix in the network.

The argument in [7] neglects two important considerations. First, the relative cost of IMP buffer storage and 50Kb circuits is such that one should probably not try to optimize IMP buffer storage. The true trade-off which governs packet size might well be efficient use of phone line bandwidth (driving packets larger) vs. delay characteristics (driving packets smaller). If buffer storage is limiting, perhaps one should just buy more.

Second, given that the IMP has sufficient buffering to handle the phone lines, one can see that this buffer space will often go unused. In fact, since there is a direct relation between storage needed and bandwidth supported, the buffers will go unused except for the case of high bandwidth utilization—i.e., file transfers. But in this case one would hope the user is sending multi-packet messages. In short, the IMP buffers are set up to handle file transfers—the hard case—and coast along under the easy interactive cases currently prevalent in the net.

It might be possible to saturate an IMP with tiny packets—although it is hard to imagine enough interactive users to do so. If so, the overhead one is paying for the space on the phone lines is so enormous* that there would be tremendous incentive to pack the tiny messages up into longer units—and we are back to the correct operating point. In summary: being busy implies use of large packets which implies full buffers; when idle the buffer size does not matter.

As is implicit in the above, the choice of packet size is influenced by many factors. Since some of the factors are inherently in conflict, making an optimum difficult to define, much less find, the current ARPA Network packet size is a good compromise. Other packet sizes (e.g., 2000 bits) would also probably work, but there is certainly no alternative choice for packet size which has clear superiority over the l000-bit size currently used.

5. Lockups, Interference, and Other Bugs and Shortcomings

We next turn to a brief point-by-point examination of some flaws and apparent flaws in the ARPA Network. There are problems that can be described as lockups, others which are inefficiencies due to congestion or interference, and others which might be called bugs, oversights, or shortcomings.

One point made by Cerf in [4] is that the system will lead to lockup "because the destination IMP will accept packets which arrive out of order and buffers them until they can be reordered for transmission to the destination Host". This simply is not true. The IMP only accepts packets for which storage has been allocated, or for which storage can be allocated. Since it honors requests for allocation strictly in message number sequence, no lockups can result.

The apparent problem pointed to by Cerf is related to the bug noted by Kleinrock and Opderbeck in [8]. Here if a single-packet message arrives out of order at the destination IMP as a request for storage, in the midst of a series of multi-packet messages, a lockup could result unless the IMP follows this simple rule: a multi-packet allocate must never use up the last free buffer on the IMP. With this rule (subsequently installed in the ARPA Network), there is always room for a single-packet allocate to be returned. With this breathing space, the IMP is able to deal with arbitrarily many single-packet messages which arrive out of order. It can be noted here that the ease with which it was possible to fix this bug (which had never occurred in practice) is a strong

* Up to a factor of 100 times the true data rate.

argument for keeping the message processing functions in the IMP. We should also mention that this does not represent a flaw in the storage allocation concept but rather a bug in our implementation.

Another problem discovered by Opderbeck [9] is also related to the interaction of the sequencing and flow control mechanisms. This shortcoming of the system does not lead to a lockup, but rather to degraded performance due to retransmissions. Opderbeck shows that if an IMP is sending a continuous stream of single-packet messages to another IMP, and one arrives out of order, it is necessary for the destination IMP to return an ALLOCATE instead of RFNM message. Subsequently, as long as the traffic persists, all messages will appear to be out of order, since they arrive at the destination prior to the retransmitted copy of the previous message. Thus all messages will go through an allocate/retransmission cycle, cutting bandwidth and increasing delay. Several approaches are possible to solve this inefficiency; one is simply to allocate a few buffers in the IMP for accepting out-of-sequence messages, in order to escape from this syndrome when it occurs. To date we have implemented a temporary fix, but we have postponed a final solution to this problem until it is better understood.

A different kind of problem has been noted by us and others in the ARPA Network as a more serious and present difficulty. In exchanging the earlier idea of a link table for a message number table between each pair of IMPs, we coupled the Hosts at an IMP quite tightly. That is, if two Hosts at IMP *A* communicate with two Hosts at IMP *B*, they share the same message number sequence. This means they can interfere with each other, if their rates are very different. Further, neither Host has access to the full window of four messages in flight. We knew about this situation when we installed the tables two years ago, and felt it was just one instance of the inescapable fact that all of an IMP’s resources are shared among the Hosts to which it is connected, and that it was not worth the extra storage to decouple their message numbers. In the time since that change, the Network has grown dramatically, and we have come to feel that any unnecessary or artificial coupling of the Hosts at an IMP is undesirable. In fact, we are in the process of putting in several explicit measures to provide fairness in resource allocation among the Hosts as discussed in our Quarterly Technical Report No. 6. Foremost among these is a change to allow separate message number sequences for each Host-to-Host pair. Other measures will include checks on the use of storage at the source and destination IMPs so that no one Host can get an unfair share of buffering if there are other Hosts which also need it. We will also continue to organize the processing of messages in the IMP so that no Host has priority over any other.

Another complaint, somewhat less justified in our view, is that true restriction to four messages in flight between two IMPs is limiting the performance of some Hosts. In the case of 8000-bit messages, and a round-trip delay of 1/2 second, this allows a throughput of 64Kbs, which is not unreasonable given the current ARPA Network. The number of messages in flight cannot be extended indefinitely, since the IMPs provide buffering by allocating storage for the messages. Also, there is a significant amount of bookkeeping associated with each message. Nevertheless, we have decided to extend the message number window to eight at the time that we change the system to maintain independent message numbers for each Host pair.

Some Hosts have stated that one aspect of the IMP-Host interface is undesirable, namely that the IMP is free to stop accepting bits from the Host whenever it is not finished processing a message. This happens, for instance, when the source IMP requests storage from the destination IMP for a multi-packet message. No other messages can be sent during this period, since the source IMP deals with one message at a time. The same situation exists when the source IMP needs to acquire a message number when none are free, or to find a free source buffer, and so on.

Hosts which serve one network process have no difficulty here, but Hosts with many users, time-sharing systems for example, would prefer never to have their Host-to-IMP interface stopped. They may have other users to service while the IMP is acquiring the resources necessary to deal with an earlier message. This situation could be corrected by offering an optional message type from the Host to the IMP which says, in effect, “Please take this message now, or else discard it and tell me when you can take it.” Alternatively, we could allow the Host to send the IMP a message which is a notice of intent to send, which the IMP will answer with a permit to send, guaranteeing that the IMP will never refuse to take Host messages.

Yet another problem for some Host applications, related to the Host being stopped, is the 30-second incomplete transmission timeout on message numbers. Some Hosts need to have accurate accounting of all messages, while others would prefer to go ahead, giving up on a lost message or reply in much less than 30 seconds. That is, they are willing to trade off a higher rate of lost messages, declaring messages older than a few seconds as lost, for the ability to always send a message every few seconds. Here again, we are considering allowing the Hosts to specify their preference in this regard, once message number sequences are kept separately for each Host pair.

An example of a Host application with special performance requirements is the transmission of real-time, synchronous data such as vocoded speech, the output of physical sensing devices, and so on. These users represent a new demand for network performance—low delay and high throughput at the same time. The current ARPA Network is optimized to deliver short messages with low delay, and long messages with high throughput. Speech transmission, for instance, requires a “guaranteed” minimum throughput level (depending on the vocoding technique), and a “guaranteed” maximum delay (depending on the tolerance of the people speaking). Users are probably willing to trade a slightly higher message loss rate for the “guarantees”. This is a good example of an environment in which reliability, in terms of accurate reporting of message loss, is less important than steady high performance. Of course, the reverse is true in transferring a data file from one system to another. It has been argued that the requirements of such applications as speech are so stringent that the network should do no message processing at all, to avoid introducing artificial performance limits, and leave these functions to the Hosts. We feel this argument is specious, for many of the general reasons stated above about the question of message processing in the subnetwork. Furthermore, we believe we can modify the algorithms in the subnetwork to provide appropriate service to such Hosts, without prejudicing the network services provided to other Hosts (as would happen if the subnetwork abandoned all message processing).

Another area of interest is network interconnection. It can be argued that (1) message processing facilities offered by the several interconnected networks vary greatly and there may be no need for elaborate processing in one network if it is not provided in another; and (2) some functions, such as sequencing, might lead to degraded overall performance if performed in each network (total delay is increased if each net in the path performs sequencing in turn instead of passing unordered data to the ultimate destination for sequencing). We have several reactions to these arguments:

- There are not yet very many packet networks. For all the reasons discussed above, new networks which avoid subnetwork ordering may have more problems than they expect. Some operating experience should perhaps be obtained before concluding that subnetwork ordering is easy to avoid.

- If networks are interconnected where some nets perform ordering on multi-packet messages and other nets do not, it is still feasible to deal with single-packet internetwork traffic without incurring degradation.

- Some networks may not function well if services are removed which do not exist in other networks. For instance, flow control might be important to keep in one network even though some other network does not perform flow control.

- In the ARPA Network case, if it is politically or technically desirable to avoid some of the standard message processing for messages that are destined for internetwork transmission, it is relatively easy to allow parallel mechanisms which are subsets of the full message processing facilities. (We are currently implementing such a parallel mechanism to permit experiments of this type).

A different kind of Host application is one that is highly sensitive to errors of any kind, such as program or data file transfers. For these Hosts, a scheme such as end-to-end checksumming might be appropriate, if the cost were not too high. The IMP might be able to interact with such a scheme, to the minimal extent of simply passing a Host-generated checksum along, or to a greater extent by generating and verifying checksums for the Host.

There is a potential use of the ARPA Network which, though desirable from some points of view, is difficult to implement at present. That is the connection of a Host to more than one IMP, for the purpose of increasing reliability. Improving performance by using multiple connections simultaneously is also a possibility. The question of how and where to translate a Host’s logical name into one of its several physical addresses can be answered in several ways; we think the subnetwork could easily perform this translation as an extension of the current routing algorithm. Dynamic translation tables could be maintained at all IMPs. A much more difficult problem concerns the message processing for that Host. How can sequencing be performed if different messages (or even packets) of a data stream are sent to different physical addresses? We do not have a good answer as to how to maintain multiple connections of a Host to the Network while maintaining sequencing, error control, and so on. Either these message processing facilities would have to be disabled, or else only one Host connection could be active at any one time, or the translation from physical to logical address would have to be performed by the source Host. Of course, with Host-level ordering, multiple connections pose no additional problems.

The final topic to consider under the general heading of problems with the current ARPA Network approach is that of adapting the various techniques described for use in large networks (hundreds of IMPs, thousands of Hosts). The key point is that the cost of keeping linear tables indexed by IMP, Host, or anything else becomes prohibitive at some network size. The simplicity and reliability of such tables, and their resulting freedom from deadlock, are naturally desirable, but cannot be maintained at arbitrarily high cost. The current tables are structured for a network of up to 63 IMPs, 4 Hosts per IMP (plus 4 fake Hosts internal to each IMP), with line bandwidth of 50Kbs typically, and round-trip delays of roughly 1/2 second. With the growth of the ARPA Network, and the introduction of new technology in IMPs and circuits, including satellites, all the parameters above must be reexamined.

REFERENCES

1. “Basic Elements of a Network Data Link Control Procedure (NDLC),” L. Pouzin, INWG 54, NIC 30375, January 1974.

2. “Presentation and Major Design Aspects of the Cyclades Computer Network,” L. Pouzin, *Proceedings of the Third ACM Data Communications Symposium*, November 1973, pp. 80–88.

3. “A Protocol for Packet Network Intercommunication,” V. Cerf and R. Kahn, *IEEE Transactions on Communications*, Vol. COM-22, No. 5, May 1974, pp. 637–648.

4. “An Assessment of ARPANET Protocols,” V. Cerf, RFC 635, NIC 30489, April 1974.

5. “Packet Communication,” R. M. Metcalfe, Massachusetts Institute of Technology Project MAC Report MAC TR-114, December 1973.

6. *Communication Networks for Computers*, D. W. Davies and D. L. A. Barber, London: John Wiley and Sons, 1973.

7. “On Measured Behavior of the ARPA Network,” L. Kleinrock and W. Naylor, *Proceedings AFIPS 1974 NCC*, Vol. 43, pp. 767–780.

8. “On a Possible Lockup Condition in the IMP Subnet Due to Message Sequencing,” L. Kleinrock and H. Opderbeck, RFC 626, NIC 22161, March 1974.

9. “Throughput Degradations for Single Packet Messages,” H. Opderbeck, RFC 632, NIC 30239, May 1974.