A Protocol for Packet Network Intercommunication

VINTON G. CERF AND ROBERT E. KAHN,
MEMBER, IEEE

Abstract — A protocol that supports the sharing of resources that exist in different packet switching networks is presented. The protocol provides for variation in individual network packet sizes, transmission failures, sequencing, flow control, end-to-end error checking, and the creation and destruction of logical process-to-process connections. Some implementation issues are considered, and problems such as internetwork routing, accounting, and timeouts are exposed.

INTRODUCTION

IN THE LAST few years considerable effort has been expended on the design and implementation of packet switching networks [1]-[7],[14],[17]. A principle reason for developing such networks has been to facilitate the sharing of computer resources. A packet communication network includes a transportation mechanism for delivering data between computers or between computers and terminals. To make the data meaningful, computer and terminals share a common protocol (i.e, a set of agreed upon conventions). Several protocols have already been developed for this purpose [8]-[12],[16]. However, these protocols have addressed only the problem of communication on the same network. In this paper we present a protocol design and philosophy that supports the sharing of resources that exist in different packet switching networks.

After a brief introduction to internetwork protocol issues, we describe the function of a GATEWAY as an interface between networks and discuss its role in the protocol. We then consider the various details of the protocol, including addressing, formatting, buffering, sequencing, flow control, error control, and so forth. We close with a description of an interprocess communication mechanism and show how it can be supported by the internetwork protocol.

Even though many different and complex problems must be solved in the design of an individual packet switching network, these problems are manifestly compounded when dissimilar networks are interconnected. Issues arise which may have no direct counterpart in an individual network and which strongly influence the way in which internetwork communication can take place.

A typical packet switching network is composed of a set of computer resources called HOSTS, a set of one or more *packet switches*, and a collection of communication media that interconnect the packet switches. Within each HOST, we assume that there exist *processes* which must communicate with processes in their own or other HOSTS. Any current definition of a process will be adequate for our purposes [13]. These processes are generally the ultimate source and destination of data in the network. Typically, within an individual network, there exists a protocol for communication between any source and destination process. Only the source and destination processes require knowledge of this convention for communication to take place. Processes in two distinct networks would ordinarily use different protocols for this purpose. The ensemble of packet switches and communication media is called the *packet switching subnet*. Fig. 1 illustrates these ideas.

In a typical packet switching subnet, data of a fixed maximum size are accepted from a source HOST, together with a formatted destination address which is used to route the data in a store and forward fashion. The transmit time for this data is usually dependent upon internal network parameters such as communication media data rates, buffering and signalling strategies, routeing, propagation delays, etc. In addition, some mechanism is generally present for error handling and determination of status of the networks components.

Individual packet switching networks may differ in their implementations as follows.

1) Each network may have distinct ways of addressing the receiver, thus requiring that a uniform addressing scheme be created which can be understood by each individual network.

2) Each network may accept data of different maximum size, thus requiring networks to deal in units of the smallest maximum size (which may be impractically small) or requiring procedures which allow data crossing a network boundary to be reformatted into smaller pieces.

3) The success or failure of a transmission and its performance in each network is governed by different time delays in accepting, delivering, and transporting the data. This requires careful development of internetwork timing procedures to insure that data can be successfully delivered through the various networks.

4) Within each network, communication may be disrupted due to unrecoverable mutation of the data or missing data. End-to-end restoration procedures are desirable to allow complete recovery from these conditions.

Paper approved by the Associate Editor for Data Communications of the IEEE Communications Society for publications without oral presentation. Manuscript received November 5, 1973. The research reported in this paper was supported in part by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC 15-73-C-0370.
V.G. Cerf is with the Department of Computer Science and Electrical Engineering, Standford University, Stanford, Calif.
R.E. Kahn is with the Information Processing Technology Office, Advanced Research Projects Agency, Department of Defense, Arlington, Va.

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

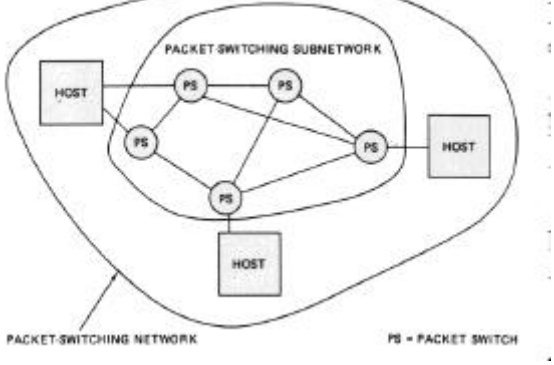

Fig. 1. Typical packet switching network.

5) Status information, routing, fault detection, and isolation are typically different in each network. thus, to obtain verification of certain conditions, such as an inaccessible or dead destination, various kinds of coordination must be invoked between the communicating networks.

It would be extremely convenient if all the differences between networks could be economically resolved by suitable interfacing at the network boundaries. For many of the differences, this objective can be achieved. However, both economic and technical considerations lead us to prefer that the interface be as simple and reliable as possible and deal primarily with passing data between networks that use different packet switching strategies.

The question now arises as to whether the interface ought to account for differences in HOST or process level protocols by transforming the source conventions into the corresponding destination conventions. We obviously want to allow conversion between packet switching strategies at the interface, to permit interconnection of existing and planned networks. However, the complexity and dissimilarity of the HOST or process level protocols makes it desirable to avoid having to transform between them at the interface, even if this transformation were always possible. Rather, compatible HOST and process level protocols must be developed to achieve effective internetwork resource sharing. The unacceptable alternative is for every HOST or process to implement every protocol (a potentially unbounded number) that may be needed to communicate with other networks. We therefore assume that a common protocol is to be used between HOST'S or processes in different networks and that the interface between networks should take as small a role as possible in this protocol.

To allow networks under different ownership to interconnect, some accounting will undoubtedly be needed for traffic that passes across the interface. In its simplest terms, this involves an accounting of packets handled by each net for which charges are

interconnection must preserve intact the internal operation of each individual network. This is easily achieved if two networks interconnect as if each were a HOST to the other network, but without utilising or indeed incorporating any elaborate HOST protocol transformations.

It is thus apparent that the interface between networks must play a central role in the development of any network interconnection strategy. We give a special name to this interface that performs these functions and call it a GATEWAY.

THE GATEWAY NOTION

In Fig. 2 we illustrate three individual networks labelled *A*, *B*, and *C* which are joined by GATEWAYS *M* and *N*. GATEWAY *M* interfaces network *A* with network *B*, and GATEWAY *N* interfaces network *B* to network *C*. We assume that an individual network may have more than one GATEWAY (e.g., network *B*) and that there may be more than one GATEWAY path to use in going between a pair of networks. The responsibility for properly routing data resides in the GATEWAY.

In practice, a GATEWAY between two networks may be composed of two halves, each associated with its own network. It is possible to implement each half of a GATEWAY so it need only embed internetwork packets in local packet format or extract them. We propose that the GATEWAY handle internetwork packets in a standard format, but we are not proposing any particular transmission procedure between GATEWAY halves.

Let us now trace the flow of data through the interconnected networks. We assume a packet of data from process *X* enters network *A* destined for process *Y* in network *C*. The address of *Y* is initially specified by process *X* and the address of GATEWAY *M* is derived from the address of process *Y*. We make no attempt to specify whether the choice of GATEWAY is made by process *X*, its HOST, or one of the packet switches in network *A*. The packet traverses network *A* until it reaches GATEWAY *M*. At the GATEWAY, the packet is reformatted to meet the requirements of network *B*, account is taken of this unit of flow between *A* and *B*, and the GATEWAY delivers the packet to network *B*. Again the derivation of the next GATEWAY address is accomplished based on the address of the destination *Y*. In this case, GATEWAY *N* is the next one. The packet traverses network *B* until it finally reaches GATEWAY *N* where it is formatted to meet the requirements of network *C*. Account is again taken of this unit of flow between networks *B* and *C*. Upon entering network *C*, the packet is routed to the HOST in which process *Y* resides and there it is delivered to its ultimate destination.

© 1974 IEEE. Reprinted, with permission, from IEEE Trans on Comms, Vol Com-22, No 5 May 1974

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

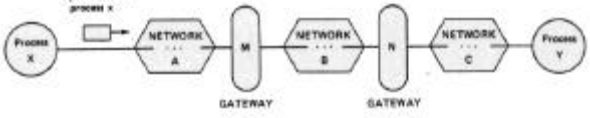

Fig. 2. Three networks interconnected by two GATEWAYS.

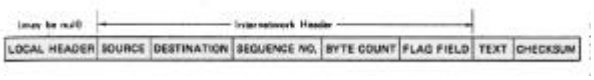

Fig. 3. Internetwork packet format (fields not shown to scale).

Since the GATEWAY must understand the address of the source and destination HOSTS, this information must be available in a standard format in every packet which arrives at the GATEWAY. This information is contained in an *internetwork header* prefixed to the packet by the source HOST. The packet format, including the internetwork header, is illustrated in Fig. 3. The source and destination entries uniformly and uniquely identify the address of every HOST in the composite network. Addressing is a subject of considerable complexity which is discussed in greater detail in the next section. The next two entries in the header provide a sequence number and a byte count that may be used to properly sequence the packets upon delivery to the destination and may also enable the GATEWAYS to detect fault conditions affecting the packet. The flag field is used to convey specific control information and is discussed in the section on retransmission and duplicate detection later. The remainder of the packet consists of text for delivery to the destination and a trailing check sum used for end-to-end software verification. The GATEWAY does *not* modify the text and merely forwards the check sum along without computing or recomputing it.

Each network may need to augment the packet format before it can pass through the individual network. We have indicated a *local header* in the figure which is prefixed to the beginning of the packet. This local header is introduced merely to illustrate the concept of embedding an internetwork packet in the format of the individual network through which the packet must pass. It will obviously vary in its exact form from network to network and may even be unnecessary in some cases. Although not explicitly indicated in the figure, it is also possible that a local trailer may be appended to the end of the packet.

Unless all transmitted packets are legislatively restricted to be small enough to be accepted by every individual network, the GATEWAY may be forced to split a packet into two or more smaller packets. This action is called fragmentation and must be done in such a way that the destination is able to piece together the fragmented packet. It is clear that the internetwork header format imposes a minimum packet size which all networks must carry (obviously all networks will want to carry packets

range growth and development of internetwork communication would be seriously inhibited by specifying how much larger than the minimum a packet size can be, for the following reasons.

1) If a maximum permitted packet size is specified then it becomes impossible to completely isolate the internal packet size parameters of one network from the internal packet size parameters of all other networks.

2) It would be very difficult to increase the maximum permitted packet size in response to new technology (e.g. large memory systems, higher data rate communication facilities, etc.) since this would require the agreement and then implementation by all participating networks.

3) Associative addressing and packet encryption may require the size of a particular packet to expand during transit for incorporation of new information.

Provision for fragmentation (regardless of where it is performed) permits packet size variations to be handled on an individual network basis without global administration and also permits HOSTS and processes to be insulated from changes in the packet sizes permitted in any networks through which their data must pass.

If fragmentation must be done, it appears best to do it upon entering the next network at the GATEWAY since only this GATEWAY (and not the other networks) must be aware of the internal packet size parameters which made the fragmentation necessary.

If a GATEWAY fragments an incoming packet into two or more packets, they must eventually be passed along to the destination HOST as fragments or reassembled for the HOST. It is conceivable that one might desire the GATEWAY to perform the reassembly to simplify the task of the destination HOST (or process) and/or to take advantage of the larger packet size. We take the position that GATEWAY should not perform this function since GATEWAY reassembly can lead to serious buffering problems, potential deadlocks, the necessity for all fragments of a packet to pass through the same GATEWAY, and increased delay in transmission. Furthermore, it is not sufficient for the GATEWAY to provide this function since the final GATEWAY may also have to fragment a packet for transmission. Thus the destination HOST must be prepared to do this task.

Let us now turn briefly to the somewhat unusual accounting effect which arises when a packet may be fragmented by one or more GATEWAY. We assume, for simplicity, that each network initially charges a fixed rate per packet transmitted, regardless of distance, and if one network can handle a larger packet size than another, it charges a proportionally larger price per packet. We also assume that a subsequent increase in any network's packet size does not result in additional cost per packet to its users. The charge to a user thus remains

© 1974 IEEE. Reprinted, with permission, from IEEE Trans on Comms, Vol Com-22, No 5 May 1974

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

packet is fragmented into smaller packets which must individually pass through a subsequent network with a larger packet size than the original unfragmented packet. We expect that most networks will naturally select packet sizes close to one another, but in any case, an increase in packet size in one net, even when it causes fragmentation, will not increase the cost of transmission and may actually decrease it. In the event that any other packet charging policies (than the one we suggest) are adopted, differences in cost can be used as an economic lever toward optimisation of individual network performance.

PROCESS LEVEL COMMUNICATION

We suppose that processes wish to communicate in full duplex with their correspondents using unbounded but finite length messages. A single character might constitute the text of a message from a process to a terminal or vice versa. An entire page of characters might constitute the text of a message from a file to a process. A data stream (e.g. a continuously generated bit string) can be represented as a sequence of finite length messages.

Within a HOST we assume that existence of a transmission control program (TCP) which handles the transmission and acceptance of messages on behalf of the processes it serves. The TCP is in turn served by one or more packet switches connected to the HOST in which the TCP resides. Processes that want to communicate present messages to the TCP for transmission, and TCP's deliver incoming messages to the appropriate destination processes. We allow the TCP to break up messages into segments because the destination may restrict the amount of data that may arrive, because the local network may limit the maximum transmissin size, or because the TCP may need to share its resources among many processes concurrently. Furthermore, we constrain the length of a segment to an integral number of 8-bit bytes. This uniformity is most helpful in simplifying the software needed with HOST machines of different natural word lengths. Provision at the process level can be made for padding a message that is not an integral number of bytes and for identifying which of the arriving bytes of text contain information of interest to the receiving process.

Mutliplexing and demultiplexing of segments among processes are fundamental tasks of the TCP. On transmission, a TCP must multiplex together segments from different source processes and produce internetwork packets for delivery to one of its serving packet switches. On reception, a TCP will accept a sequence of packets from its serving packet switch(es). From this sequence of arriving packets (generally from different HOSTS), the TCP

We assume that every segment is augmented with additional information that allows transmitting and receiving TCP's to identify destination and source processes, respectively. At this point, we must face a major issue. How should the source TCP format segments destined for the same destination TCP? We consider two cases.

Case 1): If we take the position that segment boundaries are immaterial and that a byte stream can be formed of segments destined for the same TCP, then we may gain improved transmission efficiency and resource sharing by arbitrarily parceling the stream into packets, permitting many segments to share a single internetwork packet header. However, this position results in the need to reconstruct exactly, and in order, the stream of text bytes produced by the source TCP. At the destination, this stream must first be parsed into segments and these in turn must be used to reconstruct messages for delivery to the appropriate processes.

There are fundamental problems associated with this strategy due to the possible arrival of packets out of order at the destination. The most critical problem appears to be the amount of interference that processes sharing the same TCP-TCP byte stream may cause among themselves. This is especially so at the receiving end. First, the TCP may be put to some trouble to parse the stream back into segments and then distribute them to buffers where messages are reassembled. If it is not readily apparent that all of a segment has arrived (remember, it may come as several packets), the receiving TCP may have to suspend parsing temporarily until more packets have arrived. Second, if a packet is missing, it may not be clear whether succeeding segments, even if they are identifiable, can be passed on to the receiving process, unless the TCP has knowledge of some process level sequencing scheme. Such knowledge would permit the TCP to decide whether a succeeding segment could be delivered to its waiting process. Finding the beginning of a segment when there are gaps in the byte stream may also be hard.

Case 2): Alternatively, we might take the position that the destination TCP should be able to determine, upon its arrival and without additional information, for which process or processes a received packet is intended, and if so, whether it should be delivered then.

If the TCP is to determine for which process an arriving packet is intended, every packet must contain a *process header* (distinct from the internetwork header) that completely identifies the destination process. For simplicity, we assume that each packet contains text from a single process which is destined for a single process. Thus each

© 1974 IEEE. Reprinted, with permission, from IEEE Trans on Comms, Vol Com-22, No 5 May 1974

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

destination process, the TCP must be able to determine whether the data is in the proper sequence (we can make provision for the destination process to instruct its TCP to ignore sequencing, but this is considered a special case). With the assumption that each arriving packet contains a process header, the necessary sequencing and destination process identification is immediately available to the destination TCP.

Both Cases 1) and 2) provide for the demultiplexing and delivery of segments to destination processes, but only Case 2) does so without the introduction of potential interprocess interference. Furthermore, Case 1) introduces extra machinery to handle flow control on a HOST-to-HOST basis, since there must also be some provision for process level control, and this machinery is little used since the probability is small that within a given HOST, two processes will be coincidentally scheduled to send messages to the same destination HOST. For this reason, we select the method of Case 2) as a part of the *internetwork transmission protocol*.

ADDRESS FORMATS

The selection of address formats is a problem between networks because the local network addresses of TCP's may vary substantially in format and size. A uniform internetwork TCP address space, understood by each GATEWAY and TCP, is essential to routing and delivery of internetwork packets.

Similar troubles are encountered when we deal with process addressing and, more generally, port addressing. We introduce the notion of *ports* in order to permit a process to distinguish between multiple message streams. The port is simply a designator of one such message stream associated with a process. The means for identifying a port are generally different in different operating systems, and therefore, to obtain uniform addressing, a standard port address format is also required. A port address designates a full duplex message stream.

TCP ADDRESSING

TCP addressing is intimately bound up in routeing issues, since a HOST or GATEWAY must choose a suitable destination HOST or GATEWAY for an outgoing internetwork packet. Let us postulate the following address format for the TCP address (Fig. 4). The choice for network identification (8 bits) allows up to 256 distinct networks. This size seems sufficient for the foreseeable future. Similarly, the TCP identifier field permits up to 65 536 distinct TCP's to be addressed, which seems more than sufficient for any given network.

determine how to route the packet. If the destination network is connected to the GATEWAY, the lower 16 bits of the TCP address are used to produce a local TCP address in the destination network. If the destination network is not connected to the GATEWAY, the upper 8 bits are used to select a subsequent GATEWAY. We make no effort to specify how each individual network shall associate the internetwork TCP identifier with its local TCP address. We also do not rule out the possibility that the local network understands the internetwork addressing scheme and thus alleviates the GATEWAY of the routing responsibility.

PORT ADDRESSING

A receiving TCP is faced with the task of demultiplexing the stream of internetwork packets it receives and reconstructing the original messages for each destination process. Each operating system has its own internal means of identifying processes and ports. We assume that 16 bits are sufficient to serve as internetwork port identifiers. A sending process need not know how the destination port identification will be used. The destination TCP will be able to parse this number appropriately to find the proper buffer into which it will place arriving packets. We permit a large port number field to support processes which want to distinguish between many different message streams concurrently. In reality, we do not care how the 16 bits are sliced up by the TCP's involved.

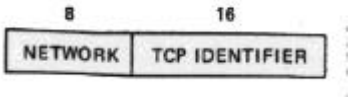

Fig. 4. TCP address.

Even though the transmitted port name field is large, it is still a compact external name for the internal representation of the port. The use of short names for port identifiers is often desirable to reduce transmission overhead and possibly reduce packet processing time at the destination TCP. Assigning short names to each port, however, requires an initial negotiation between source and destination to agree on a suitable short name assignment, the subsequent maintenance of conversion tables at both the source and the destination, and a final transaction to release the short name. For dynamic assignment of port names, this negotiation is generally necessary in any case.

SEGMENT AND PACKET FORMATS

As shown in Fig. 5, messages are broken by the TCP into segments whose format is shown in more detail in Fig. 6. The field lengths illustrated are

© 1974 IEEE. Reprinted, with permission, from IEEE Trans on Comms, Vol Com-22, No 5 May 1974

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.