# Internetwork Protocol Approaches

JONATHAN B. POSTEL

*(Invited Paper)*

*Abstract*—**The motivation for interconnecting networks is to provide one or more consistent services to the set of users of the interconnected networks. To provide these services either new end-to-end service protocols must be defined or the service protocols of the individual networks must be made to interwork. In either case the issues of addressing, routing, buffering, flow control, error control, and security must be considered. Two examples of interconnection strategy are examined: the interconnection of X.25 networks, and the interconnection of ARPA research networks. The models for interconnection of networks and the role of internetwork protocols are discussed.**

## INTRODUCTION

THE motivations for constructing computer communication networks—data and program exchange and sharing, remote access to resources, etc.—are also motivations for interconnecting networks. This follows from the observation that the power of a communication system is related to the number of potential participants.

This paper first discusses a few key concepts involved in computer communication networks. The view that computer networks provide an interprocess communication facility is presented. The datagram and virtual circuit services are compared. The interconnection device or gateway is discussed. The relation of the interconnection issues to the open systems architecture is described.

In this paper, two approaches to internetworking are characterized: the public data network system as implied by the CCITT X.75 Recommendation and the ARPA experimental internetwork. These two systems illustrate the virtual circuit and the datagram approaches to network interconnection, respectively. The vast majority of the work on interconnecting networks falls into one of these two approaches.

## INTERPROCESS COMMUNICATION

While discussing computer communication, it is useful to recall that the communication takes place at the request and agreement of processes, i.e., computer programs in execution. Processes are the actors in the computer communication environment; processes are the senders and receivers of data. Processes operate in host computers or hosts.

The protocols used in constructing the communications capability provide an interprocess communication system. Fig. 1 shows how the combination of the network and the host network interface (hardware and software) can be viewed as providing an interprocess communication system.

When a new host computer is to be connected to an existing network, it must implement the protocol layers necessary to match the existing protocol used in the network. The new host must join the network-wide interprocess communication system so the processes in that host can communicate with processes in other hosts in the network.

The interconnection of networks requires that the processes in the hosts of the interconnected networks have a common interprocess communication system. This may be achieved by converting the networks to a new interprocess communication system, by converting one or more levels of protocol to new protocols, or by translating between pairs of interprocess communication systems at their points of contact.

## DATAGRAMS AND CIRCUITS

Two types of service are commonly discussed as appropriate for the network-provided interprocess communication service: datagrams and virtual circuits.

Datagrams are one-shot simple messages. They are inherently unreliable since they travel one-way and are not acknowledged. Datagrams may also arrive in a different order than sent (at least in some networks). Datagrams are simple to implement since they do not require the networks or gateways to record and update state information. Datagrams must carry complete address information in each message. The transmission of datagrams by a process is via send and receive actions.

Virtual circuits (or connections) are designed to be reliable and to deliver data in the order sent. Implementation of virtual circuits is complicated by the need for the networks or gateways to record and update state information. Virtual circuits are created through an exchange of messages to set up the circuit; when use terminates, an exchange of messages tears down the circuit. During the data transmission phase, a short form address or circuit identifier may be used in place of the actual address. To use a virtual circuit a process must perform actions to cause the virtual circuit to be created (call setup) and terminated, as well as the actions to send and receive data.

Datagrams provide a transaction type service while virtual circuits provide a connection type service. Each of these services is needed in a general purpose communication environment. Datagrams are most efficient for transaction type information requests such as directory assistance or weather reports. Virtual circuits are useful for terminal access to interactive computer systems for file transfer between computers.

Manuscript received June 15, 1979; revised December 18, 1979. This work was supported by the Advanced Research Projects Agency under Contract DAHC15 72 C 0308, ARPA Order 2223.

The author is with the Information Sciences Institute, University of Southern California, Marina del Rey, CA 90291.

0090-6778/80/0400-0604$00.75 © 1980 IEEE

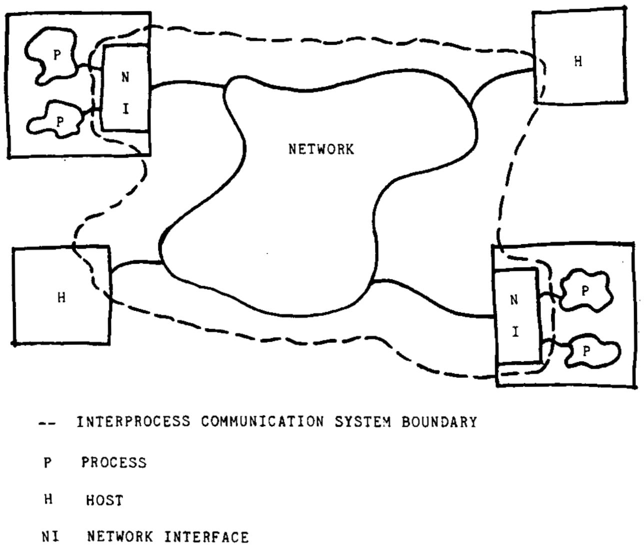


Fig. 1. Communications network.

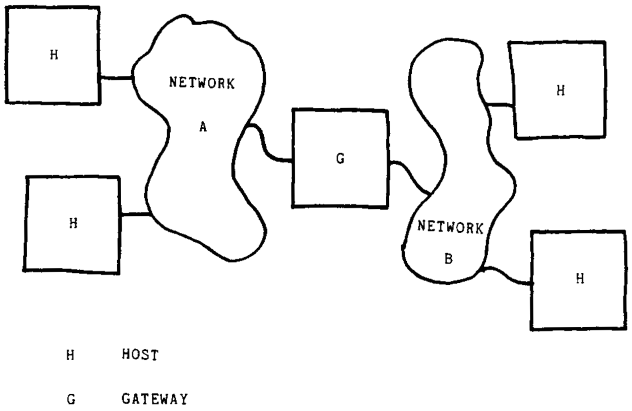


Fig. 2. Interconnected networks.

## GATEWAYS

Two or more networks are connected via a device (or pair of devices) called a gateway. Such a device may appear to each network as simply a host on that network (Fig. 2).

Some gateways simply read messages from one network (unwrapping them from that network's packaging), compute a routing function, and send messages into another network (wrapping them in that network's packaging). Since the networks involved may be implemented using different media, such as leased lines or radio transmission, this type of gateway is called a media-conversion gateway.

Other gateways may translate the protocol used in one network to that used in another network by replacing messages received from one network with different messages with the same protocol semantics sent into another network. This type of gateway is called a protocol-translation gateway.

It should be clear that the distinction between media-conversion and protocol-translation is one of degree: the media-conversion gateways bridge the gap between differing link and physical level protocols, while protocol-translation gateways bridge the gap between differing network and higher level protocols.

The translation approach to network interconnection raises several issues. Success in protocol translation seems inversely correlated with the protocol level. At the lower levels, protocol translation causes no problems because the physical level and link levels are hop-by-hop in nature. It should be noted, though, that different protocols even at these low levels may have impact on the reliability, throughput, and delay characteristics of the total communication system.

At the network and transport levels, the issues of message size, addressing, and flow control become critical. Unless one requires that only messages that can be transmitted on the network with the smallest maximum message size be sent, one must provide for the fragmentation and reassembly of messages. That is, the division of a long message into parts for transmission through a small message size network, and the reconstruction of those parts into the original message at the destination. The translation of addresses is a difficult problem when one network or transport level protocol provides a larger address space than the corresponding protocol to be translated to. When end-to-end flow control mechanisms are used, as they commonly are in transport level protocols, difficulties arise when the units controlled are different. For example, when one protocol controls octets and the corresponding protocol controls letters. More difficulties arise with potential difference in the model of flow control. For example, a difference between pre- and postallocation, or between the allocation of buffer space and the allocation of transmission rate.

At higher levels, the problems are more difficult because of the increased state information kept and the lower likelihood of one-to-one translation of individual protocol messages. A further difficulty is that each level further multiplexes the communication so that each connection or stream or channel or virtual circuit must be separately translated. It should be noted that neither of the specific interconnection approaches discussed in this paper attempts higher level protocol translation.

Gateways may be thought of as having a "half" for each network they interconnect. One could model the operation of a gateway as having each gateway-half contain procedures to convert from a network specific protocol into a standard protocol and vice versa (Fig. 3).

## RELATION TO OPEN SYSTEMS ARCHITECTURE

In relation to the open systems architecture, the interconnection of networks focuses on levels 3 and 4 [1].

To review, the open systems architecture defines the following levels of protocol:

| Level | Function |
|---|---|
| 7 | Application |
| 6 | Presentation |
| 5 | Session |
| 4 | Transport |
| 3 | Network |
| 2 | Link |
| 1 | Physical |

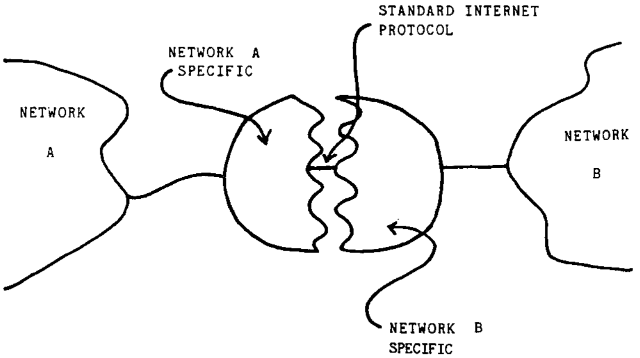


Fig. 3. Gateway halves.

The lower levels, the physical and the link levels, are hop-by-hop in nature and present no interconnection issues in terms of compatibility, although there may be some performance concerns.

The higher levels, the session level, the presentation level, and the application level, have so many compatibility requirements that it seems quite unlikely that interconnection of different protocols at those levels will be workable.

Thus, it is at the network level and the transport level that the interconnection of networks finds issues of concern.

The network level corresponds to the interface to datagram service, and the transport level corresponds to the interface to virtual circuit service.

In some networks, the network level and datagram service have been hidden from the user, forcing consideration of network interconnection at the transport level.

## INTERCONNECTION OF X.25 NETWORKS

### *Introduction*

The public data networks (PDN's) that follow the CCITT X.25 Recommendation [2] are to be interconnected via an interface specified in CCITT Recommendation X.75 [3]. Recommendation X.25 specifies the interface between the customer's equipment, called the data terminal equipment (DTE); and the network equipment, called the data circuit-terminating equipment (DCE). Recommendation X.25 implies a virtual circuit operation. Thus, the PDN's offer an interface to a virtual circuit transport level protocol. Fig. 4 shows the model of a PDN virtual circuit.

The interface between two PDN's specified in Recommendation X.75 is quite similar to that in Recommendation X.25. The equipment on either side of this interface is called a signaling terminal (STE). The STE-STE interface is much like the DTE-DCE interface. The STE-STE interconnection is a split gateway with each gateway-half in a physical device controlled by the PDN connected to that gateway-half. Fig. 5 shows the interconnection of PDN's.

The interconnection of PDN's via X.75 interfaces results in a series of virtual circuits. Each section is a distinct entity with separate flow control, error recovery, etc. Fig. 6 shows a PDN transmission path with two virtual circuits (VC's) and five separate flow control (FC) steps.

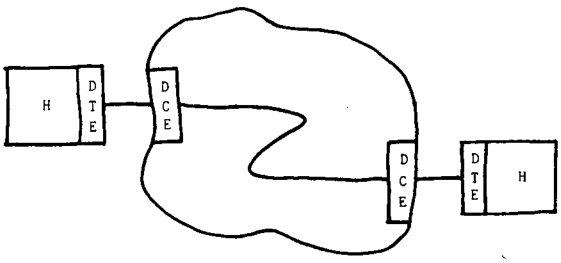


Fig. 4. PDN virtual circuit.

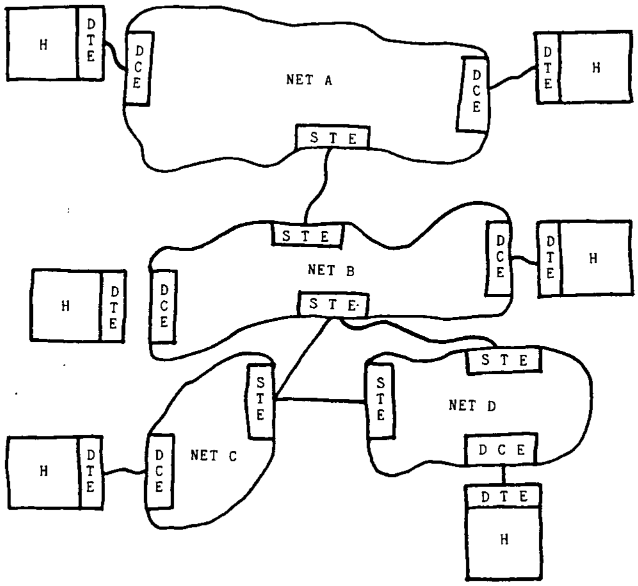


Fig. 5. Interconnection of PDN's.

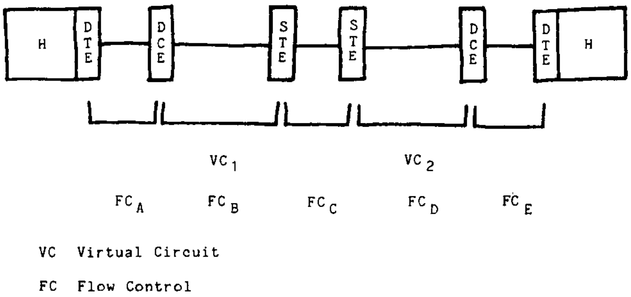


Fig. 6. PDN transmission path.

### *Addressing*

The address field is variable in length up to 15 digits, with each digit coded in a 4 bit field. The maximum address is then 60 bits (about 8 octets).

### *Routing*

The user has no influence over routing used. To create the series of virtual circuits, a series of call setups establishes a fixed route (between pairs of STE's at least). State information must be kept for each call in the source and destination DTE's and DCE's and in each STE in the route.

### *Buffering and Flow Control*

Each portion of the total path is a distinct virtual circuit. Each virtual circuit has an independent flow control (and particular to that PDN). In addition, there is flow control across each STE-STE interface. All this flow control is on a per call basis. This stepwise flow control may introduce delay in the total path that could be avoided with an end-to-end scheme.

There are some concerns about the interaction of two types of flow control implemented in PDN's. One type allows one message in transit from source DCE to destination DCE at any one time. The other allows multiple messages to be in transit, the number being determined by the flow control window.

### *Acknowledgment*

Each portion of the total path has an acknowledgment. The user to network interface also has an acknowledgment. This local acknowledgment means only that the first PDN has accepted the message for transmission, not that it has arrived at the destination.

### *Recovery*

The X.25 and X.75 Recommendations do not specify how the PDN's deal with errors internally. If unrecoverable errors occur, the network will signal a Reset, which apparently means that the virtual circuit still exists, but the flow control is reset and messages may have been lost. More serious errors result in the call being cleared.

Because of the fixed route nature of the multinetwork path, an STE failure disrupts the communication.

### *Security*

The X.25/X.75 Recommendations do not provide any security features.

### *Header Structure*

Once the call is established, a header is only 3 octets. The call setup headers are substantially longer, typically 20 octets, but possibly as large as 166 octets. There is a tradeoff between header size and state information kept; in the PDN's, the tradeoff has been made toward small headers and large state. The details of the headers are shown in Appendix I.

### *Summary*

The most important aspect of the interconnection of PDN's is that service provided to the using process is a virtual circuit with essentially the same properties a single PDN would have provided. This is done by concatenating a series of virtual circuits to provide the total path, resulting in a fixed route through a set of network interconnection points.

## INTERCONNECTION OF ARPA RESEARCH NETWORKS

### *Introduction*

The ARPA sponsored research on interconnections of networks has let to a two-level protocol to support the equivalent function of the PDN's X.25/X.75 service. The ARPA sponsored work on networks has developed an internet protocol (IP) [4], and a transmission control protocol (TCP) [5].

TCP is a logical connection transport protocol and is a level 4 protocol in the OSA model of protocol structure. The IP is a datagram protocol. The collection of interconnected networks is called an internet. IP is the network protocol of the internet and this is a level 3 protocol in the OSA model. The actual networks used are of various kinds (e.g., the ARPANET, radio networks, satellite networks, and ring or cable networks) and are referred to as local networks even though they may span continents or oceans. The interface to a local network is a local network protocol or LNP. Fig. 7 shows the model of an end-to-end connection.

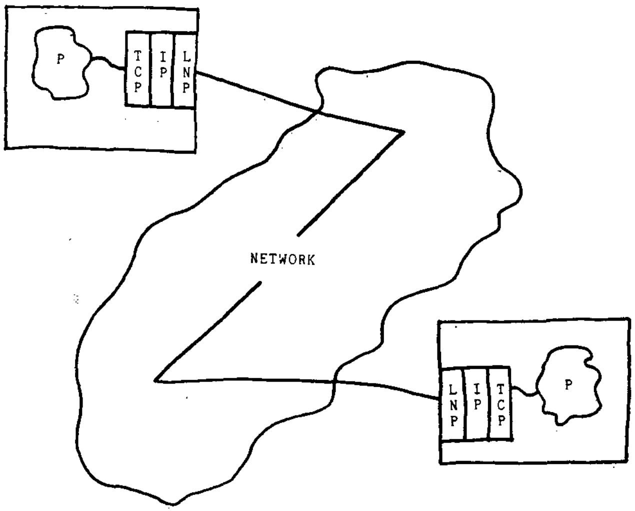


Fig. 7. End-to-end connection.

In the ARPA model, the networks interconnect via a single device called a gateway. A gateway is a host on two or more networks. Fig. 8 shows the ARPA model of the interconnection of networks.

Each network addresses a gateway on it in the same way it addresses any other host on it. The information required to deliver a message to a destination in the internet is carried in the IP header. The IP is implemented in the gateways and in hosts. A sending host prepares a datagram (which is an IP header and the original message) and then selects a gateway in its own net to forward the datagram. The sending host then sends the datagram wrapped in a local network packet to that gateway.

A gateway receives a packet from one of the local networks to which it is attached, and unwraps the IP datagram. The gateway then examines the IP header and determines the next gateway (or destination host) address in one of the local networks it is directly connected to. The gateway then sends the datagram with its IP header in a new local net packet to that gateway (or host).

The IP has no provision for flow control or error control on the data portion of the message (the IP headers are checksummed). There are no acknowledgments of IP messages. The IP is simple and the gateway may be implemented in small machines. A key point is that a gateway has no state information to record about a message. At the IP level, there are no connections or virtual circuits.

The IP does not provide a service equivalent to the PDN's X.25/X.75. To provide that type of end-to-end reliable ordered delivery of data the ARPA internet uses TCP.

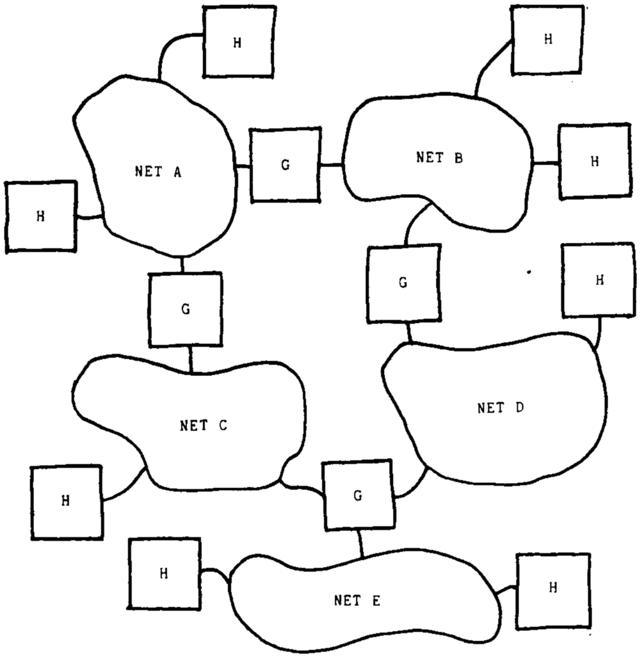


Fig. 8. ARPA model of interconnection of networks.

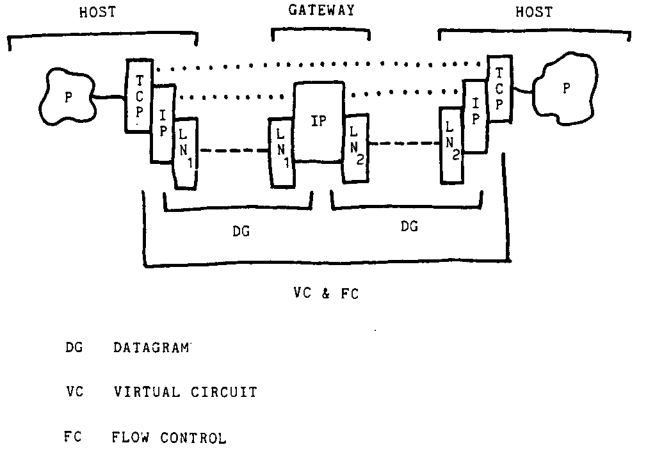


Fig. 9. ARPA model of transmission path.

TCP uses end-to-end mechanisms to ensure reliable ordered delivery of data over a logical connection. It uses flow control, positive acknowledgments with time out and retransmission, sequence numbers, etc., to achieve these goals. Fig. 9 shows the conceptual transmission path in this interprocess communication system, pointing out the datagram (DG) path between the IP modules and the virtual circuit path between the TCP modules at the source and destination and the flow control (FC) at that level.

ARPA has used these techniques to interconnect several very different networks including the ARPANET, packet radio nets, a satellite net, and several local networks.

### *Addressing*

The size of the address in this experimental system is fixed. The IP provides a one octet network field and a three octet host field. Also a one octet protocol identifier in the IP header may be considered address information. The TCP provides a two octet port field. The total of the address length is then seven octets. Provision has been made for a host to have several addresses, so the host field is sometimes called the logical host field. The total address is the concatenation of the network, host, protocol, and port fields.

### *Routing*

Normally, the user has no influence over the route used between the gateways. There is no call setup and the route may vary from one message to the next. No state information is kept in the gateways.

A user might insert a source routing option in the IP header to cause that particular message to be routed through specific gateways.

### *Buffering and Flow Control*

There is no flow control mechanism in the IP. The gateways do not control the flow on connections for they are unaware of connections or any relation between one message and the next message. The gateways may protect themselves against congestion by dropping messages. When a gateway drops a message because of congestion, it may report this fact to the source of the message.

The TCP uses end-to-end flow control using windows on a per logical connection basis.

### *Acknowledgment*

The IP has no provision for acknowledgments. The TCP uses acknowledgments for both error control and flow control. The TCP acknowledgments are not directly available to the user.

### *Recovery*

Errors in a network or gateway result in a message being dropped, and the sender may or may not be notified. This inherent unreliability in the IP level allows it to be simple and requires the end-to-end use of a reliable protocol.

TCP provides the reliable end-to-end functions to recover from any lost messages. The TCP uses a positive acknowledgment, time out, and retransmission scheme to ensure delivery of all data. Each message is covered by an end-to-end checksum.

Because of the potential of alternate routing, the end-to-end communication may be able to continue despite the failure of a gateway.

### *Security*

The IP provides an option to carry the security, precedence, and user group information compatible with AUTODIN II. The enforcement of these parameters is up to each network, and only AUTODIN II is prepared to do so.

The TCP end-to-end checksum covers all the address information (source and destination network, host, protocol, and port), so if the checksum test is successful the address fields have not been corrupted.

### Header Structure

The IP header is 20 octets (plus options, if used), but there is no call setup and no gateway state information. Thus, at the IP level, the header size versus state information tradeoff has been made toward large header and little (no) state information.

The TCP header is 20 octets (plus option, if used). There is a connection establishment procedure called the "three-way handshake," and significant state information is kept. In this case, there are both large headers and large state tables. The details of the headers are shown in Appendix II.

### Summary

The ARPA networks are interconnected by using a common datagram protocol to provide addressing (and thus routing) information and an end-to-end transport protocol to provide reliable sequenced data connections.

This model has evolved from the ARPANET experience, in particular from the internetwork protocol model suggested in a paper by Cerf and Kahn [6].

## CONCLUSION

Both the PDN's and the ARPA networks are interconnected by establishing standard protocols. The PDN's provide a virtual circuit service by concatenating the virtual circuit services of the individual networks. The ARPA networks use two levels of protocol to provide both datagram and virtual circuit services.

Additional discussion of the interconnection of PDN's is provided in [7], [8]. In another paper in this issue Boggs *et al.* present in detail another example of network interconnection using the datagram approach [9].

The issues of network interconnection have been discussed for at least 5 years (for example, McKenzie [10]). The recent expositions by Sunshine [11], Cerf and Kirstein [12], and Gien and Zimmermann [13], are particularly recommended.

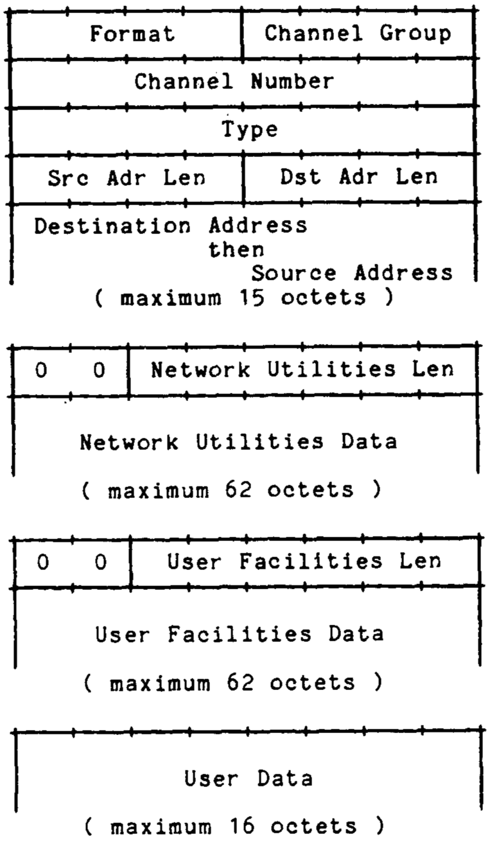
## APPENDIX I

## X.75 HEADER FORMATS

The call request and the data packet formats are illustrated here. These typify the X.75 packet formats. All the X.75 packets are the same in the first two octets. The format field indicated the type of packet.

### Call Request

The call request packet is variable in length from a practical minimum of 11 octets to an unlikely maximum of 160 octets.



*Data*

The Data packet has a three octet header.

| | |
|---|---|
| Format | Channel Group |
| Channel Number | |
| Flow Control | |
| Data | |

## APPENDIX II

## ARPA PROTOCOL HEADER FORMATS

Every datagram carries the basic IP header. Every TCP segment transmitted carries the basic TCP header.

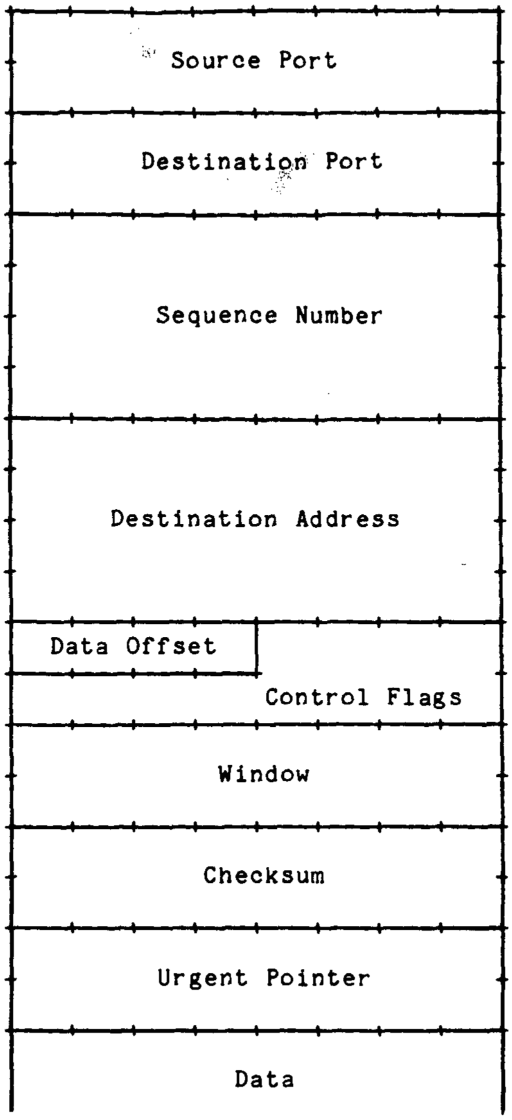
*Internet Protocol*

The ARPA IP has a basic header of 20 octets, and may carry a variable number of options up to a total length of 60 octets.

| | | |
|---|---|---|
| Version | Header Length | 1 |
| Type of Service | | 2 |
| Total Length | | 3<br>4 |
| Identification | | 5<br>6 |
| Flags | Fragment Offset | 7<br>8 |
| Time to Live | | 9 |
| Protocol | | 10 |
| Checksum | | 11<br>12 |
| Source Address | | 13<br>14<br>15<br>16 |
| Destination Address | | 17<br>18<br>19<br>20 |
| Data or TCP Header | | |

*Transmission Control Protocol*

The basic TCP header is 20 octets, and the header may be up to 60 octets long if options are used.



## REFERENCES

[1] H. Zimmermann, "The ISO reference model," this issue, pp. 425–432.

[2] "Recommendation X.25/Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode on public data networks," in *CCITT Orange Book,* vol. 7, Int. Telephone and Telegraph Consultative Committee, Geneva, Switzerland.

[3] "Proposal for provisional Recommendation X.75 on international interworking between packet switched data networks," in CCITT Study Group VII Contribution no. 207, Int. Telephone and Telegraph Consultative Committee, Geneva, Switzerland, May 1978.

[4] DARPA, "DOD standard internet protocol," IEN-128, Defense Advanced Research Projects Agency, Jan. 1980.

[5] DARPA, "DOD standard transmission control protocol," IEN-129, Defense Advanced Research Projects Agency, Jan. 1980.

[6] V. Cerf and R. Kahn, "A protocol for packet network intercommunication," *IEEE Trans. Commun.*, vol. COM-22, pp. 637–648, May 1974.

[7] G. Grossman, A. Hinchley, and C. Sunshine, "Issues in international public data networking," *Comput. Networks,* vol. 3, pp. 259–266, Sept. 1979.

[8] V. DiCiccio, C. Sunshine, J. Field, and E. Manning, "Alternatives for interconnection of public packet switching data networks," in *Proc. Sixth Data Commun. Symp.*, ACM/IEEE, Nov. 1979, pp. 120–125.

[9] D. Boggs, J. Shoch, E. Taft, and R. Metcalfe, "Pup: An internetwork architecture," this issue, pp. 612–624.

[10] A. McKenzie, "Some computer network interconnection issues," in *Proc. Nat. Comput. Conf.*, AFIPS, 1974, pp. 857–859.

★

**Jonathan B. Postel** received the B.S. and M.S. degrees in engineering and the Ph.D. degree in computer science from the University of California, Los Angeles.

He has worked for the MITRE Corporation in McLean, VA, and SRI International in Menlo Park, CA. At UCLA he was involved in the development of the ARPANET Network Measurement Center and the installation of the first host on the ARPANET. Since that time, he has participated in the development of many of the higher level protocols used in the ARPANET. He is currently a Computer Scientist at the USC/Information Sciences Institute in Marina del Rey, CA, where his research focuses on the interconnection of computer networks.