The ARPA Internet Protocol

Jonathan B. Postel, Carl A. Sunshine and Danny Cohen

Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, California 90291, USA

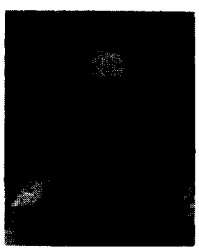
A variety of computer networks are interconnected by gateway computers in the ARPA internetwork system. Processes on different networks may exchange messages with each other by means of an Internet Protocol which must be implemented in each subscriber (host) computer and in the gateways. The Internet Protocol is a relatively simple protocol that provides for the delivery of individual messages (datagrams) with high but not perfect reliability. This Internet Protocol does not replace the existing protocol in any network, but is used by processes to extend the range of communications. Messages in Internet Protocol are transmitted through any individual network by encapsulating them in that network's protocol. This paper presents an overview of the Internet Protocol and the operation of the gateway computers in the ARPA internet system.

Keywords: Protocol, ARPA Net, Internetwork, Datagram, Gateway.

1. Introduction

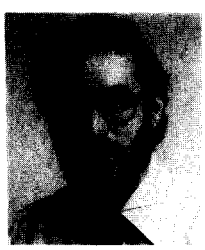
The family of computer networks developed for the United States Defense Advanced Research Projects Agency (DARPA) represents one of the largest and most diverse internetwork systems currently in operation. The basic approach to interconnecting this variety of networks was developed over several years, and has resulted in the definition of an Internet Protocol (IP) [1]. This paper is intended primarily to document the details of the IP in the open literature, and secondarily to provide a brief discussion of the major design tradeoffs which caused the IP to take its current form.

Section 2 presents an overview of the DARPA approach to interconnection and the operation of IP. Section 3 details IP's main features, while some additional options are treated in section 4. Section 5 summarizes the IP and other functions performed in the *gateways* which interconnect networks. Section 6 discusses the major design choices in developing IP. Section 7 outlines several questions and extensions requiring further work.



Jonathan Postel received his B.S. and M.S. degrees in Engineering and his Ph.D. in Computer Science from the University of California, Los Angeles. Dr. Postel has worked for the MITRE Corporation in McLean, Virginia and SRI International in Menlo Park, California. He is currently a Computer Scientist at the University of Southern California Information Sciences Institute in Marina del Rey, California. 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. His current research focuses on the interconnection of computer networks.

This research is supported by the Advanced Research Projects Agency under Contract No. DAHC15 72C 0308, Arpa Order No. 2223. The views and conclusions in this study are the authors' and should not be interpreted as representing the official opinion or policy of ARPA, the U.S. Government or any other person or agency connected with them.



Carl Sunshine is with the University of Southern California Information Sciences Institute where he is engaged in research on computer networks and their protocols. He is particularly interested in protocol modeling, and analysis, and network interconnection. Sunshine received a PhD in computer science from Stanford University in 1975, and worked at the Rand Corporation from 1975 to 1979. He is active in IFIP TC6.1 (the Internetwork Working Group) and ISO SC16, is Secretary/Treasurer of ACM SIGCOMM, and serves on the editorial board of Computer Networks journal.

Dan Cohen was born in Haifa, Israel, in 1937. He received the B.Sc. degree in mathematics from the Technion-Israel Institute of Technology in 1963, and the Ph.D. degree in computer science from Harvard University, Cambridge, MA, in 1969.

Since 1969, he has been on the faculty of Harvard University, Technion-Israel Institute of Technology, and the California Institute of Technology, Pasadena. In 1973, he joined the Information Sciences Institute of the University of Southern California, Los Angeles, here he leads several computer network-related projects. His research interests include interactive realtime systems, graphics voice communication, networking and computer architecture.

North-Holland Publishing Company
Computer Networks 5 (1981) 261–271

2. Overview

Since the development of the ARPANET in the early 1970's, a variety of new packet switching network technologies and operational networks have been developed under DARPA sponsorship, including satellite, packet radio, and local networks. In order to allow processes on different networks to communicate with each other, a means for interconnecting networks has been developed without requiring changes to the internal operation of any network.

The method chosen for interconnecting networks makes minimal demands on individual networks. To facilitate inclusion of a wide variety of networks, each net is required to provide only a minimal *datagram* level of service (i.e. to deliver individual packets of moderate length between its users with high but not perfect reliability). Networks are inter-connected by gateway computers that appear to be local subscribers on two or more nets. The gateways are responsible for routing traffic across multiple networks, and for forwarding messages across each net using the packet transmission protocol in each network. The gateways provide a point-to-point internet datagram service by concatenating the datagram services available on each individual net. Such a system of interconnected networks has been called a *Catenet* [2].

This approach allows the interconnection of networks that have significantly different internal protocols and performance. The networks in the ARPA-Catenet were originally designed as independent entities. In the Catenet approach no changes are required in the internal functions of any network.

Gateways provide an internet service by means of an Internet Protocol (IP) that defines the format of internet packets and the rules for performing internet protocol functions based on the control information (internet header) in these packets. IP must be implemented in host computers (subscribers) engaged in internet communication as well as in the gateways. Gateways also use a gateway-to-gateway protocol to exchange routing and control information.

IP provides for transmitting datagrams from an internet source to an internet destination, potentially in another net. IP also provides for *fragmentation* and *reassembly* of long datagrams, if necessary, for transmission through networks with small packet size limits.

IP is purposely limited in scope to provide only the function necessary to deliver datagrams over an interconnected system of networks. The functions of flow control, sequencing, additional data reliability, or other services commonly found in host-to-host protocols, and multidestination delivery capability or other services are purposely left for higher level protocols to provide as necessary. This allows the higher levels to be tailored to specific applications, and allows a simple and efficient implementation of IP.

2.1. Place in Protocol Hierarchy

As described above, IP functions on top of, or uses, the packet transmission protocol in each individual network. IP is used by higher level end-to-end protocols such as a reliable transport protocol, e.g., Transmission Control Protocol (TCP) [3] in the ARPA-Catenet or a "real time" protocol, e.g., for packet speech.

As shown in Figure 1, IP is the only level in the protocol hierarchy where a single common protocol is used. By locating this point of convergence at the internet datagram level, the Catenet approach preserves the flexibility to incorporate a variety of individual networks and protocols providing packet transmission below IP, while remaining general and efficient enough to serve as a common basis for a variety of higher level protocols. With this approach,

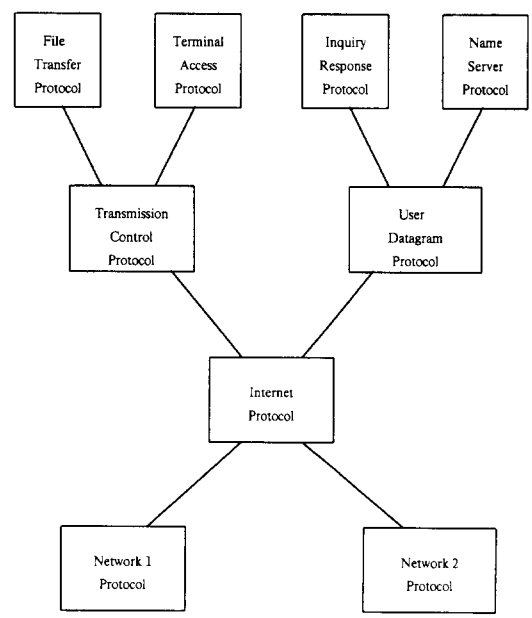


Fig. 1. Protocol Hierarchy.

gateways need only provide datagram service, and remain relatively simple, inexpensive, and efficient.

2.2. Model of Operation

The Internet Protocol provides two major functions: routing a datagram across successive networks to its internet destination address, and fragmentation/reassembly of large packets when needed to cross nets with small packet size limits. To accomplish this, an IP module must reside in each host engaged in internet communication and in each gateway that interconnects networks. The following scenario describes the progress of a datagram from source to destination (assuming one intermediate gateway is involved-see Figure 2).

The basic notion is *encapsulation.* The data to be transmitted must pass through a variety of network environments. To do this the data is encapsulated in an internet datagram. to send the datagram through an individual network, it is in turn encapsulated in a local network packet, and extracted at the other side of that network where it is decapsulated from the first network protocol and is encapsulated in the second network protocol. Thus the model is a series of encapsulation/extractions, not translations. This encapsulation is an information preserving transformation, all the information is preserved even if the individual network cannot make use of it.

The sending internet user (typically a higher level protocol module such as TCP) prepares its data and calls on its local IP module to send the data as a datagram, passing the destination address and other parameters as arguments of the call.

The IP module encapsulates the data in a datagram and fills in the datagram header. The IP module examines the internet destination address. If it is on the same network as this host, it sends the datagram directly to the destination. If the datagram is not on the same network then the IP module sends the datagram to a gateway for forwarding. The selection of which gateway to send the datagram to is an internet routing decision.

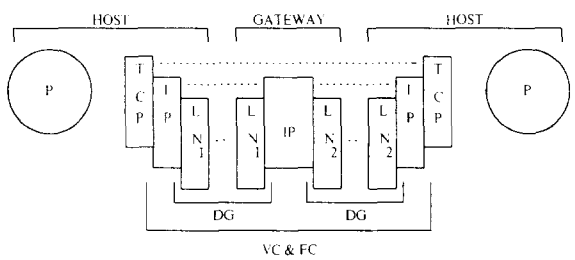


Fig. 2. ARPA Model Transmission Path.

The local network interface (note that from the IP point of view, all actual networks are "local" even if they span across the world) creates a local network packet with its own header, and encapsulates the datagram (complete with internet header) in it, then sends the result via the local network.

The datagram arrives at a gateway host encapsulated in the local network packet. The local network interface extracts the IP datagram and turns it over to the IP module.

The IP module determines from the internet destination address that the datagram should be forwarded to another host in a second network. The IP module uses the local portion of the destination address to determine the local net address for the destination host. It calls on the local network interface for the second network to send the datagram to that address.

If the datagram is too large to be sent through the second network, the IP module fragments it into several smaller datagrams and passes each one to the local net interface.

The local network interface creates a local network packet and encapsulates the datagram, sending the result to the destination host. At the destination host, the datagram is extracted from the local net packet and passed to the IP module.

The IP module determines that the datagram is for an internet user in this host. If the datagram is a fragment, the IP module collects all fragments of a particular datagram and reassembles the complete original datagram. It then passes the data to the user along with the internet source address and other information from the internet header.

2.3. Additional Mechanisms

In addition to the basic addressing and fragmentation functions described above, IP uses four key mechanisms in providing its service: *Type of Service, Time to Live, Options,* and *Header Checksum.* Each of these is summarized here and fully described in Sections 2 and 3.

The Type of Service (TOS) is used to indicate the quality of the service desired – this may be thought of as selecting among Interactive, Bulk, or Real Time, for example. The type of service is an abstract or generalized set of parameters which characterize the

service choices provided in the networks that make up the Catenet. This type of service information is used by gateways to select the actual parameters for transmission through each individual network.

The Time to Live (TTL) is an indication of the lifetime of a datagram. Datagrams must not be allowed to persist in the ARPA-Catenet indefinitely. This is because reliable end-to-end protocols depend on there being an upper bound on datagram lifetime, especially old duplicates due to retransmissions. The time to live can be thought of as a self-destruct time limit.

The Options provide for control functions useful in some situations but unnecessary for the most common communications. The options include provisions for timestamps, error reports, and special routing.

The Header Checksum provides a verification that the information used in processing the datagram has been transmitted correctly. However, the data is not covered by the checksum, and may contain errors (see Section 2.6). If the header checksum fails, the internet datagram is discarded by the entity which detects the error.

2.4. Relation to Other Work

The current ARPA Internet Protocol evolved from ideas suggested by Cerf and Kahn [4], and from contemporaneous proposals within the International Federation for Information Processing (IFIP) Technical Committee 6.1 (also known as the International Network Working Group or INWG), in which internet functions and reliable transport functions were combined in a single protocol. Subsequent development of other high level protocols (such as packet speech) that needed internet services led to splitting internet functions and reliable transport functions into separate protocols (the current IP and TCP).

The Internet Protocol used in the ARPA-Catenet is quite similar in philosophy to the PUP protocol [5] developed by the Xerox Corporation. The PUP protocol does not include fragmentation (leaving this to each local net to perform if necessary), but does include a third level of addressing (Ports within hosts) in the internet packet header. IP and PUP share the important principle of having a single common internet datagram protocol as a point of convergence in their protocol hierarchies. Both the PUP and IP systems use the encapsulation technique, and a scheme for "mutual encapsulation" has been worked out [6]. PUP and IP both trace their roots to a joint XEROX-DARPA project at Stanford University. The network interconnection approach used by the European Informatics Network [7] is also quite similar.

Public packet switching networks, on the other hand, have chosen to use virtual circuit (VC) level of service as the level of interconnection, providing end-to-end service as a concatenation of VCs through each network. Since gateways must participate at the VC level, they are more complex and costly, and the end-to-end service may be less efficient and less robust. They are also unable to accommodate "transaction" type users without setting up a VC, although the CCITT is currently considering adding a datagram type of service. For further comparison of CCITT and Catenet approaches see [8–12].

In summary, the ARPA Internet Protocol supports delivery of datagrams from an internet source to a single internet destination. IP treats each datagram as an independent entity unrelated to any other datagram. There are not connections or logical circuits (virtual or otherwise). There are no acknowledgements either end-to-end or hop-by-hop. There is no error control for data, only a header checksum. There are no retransmissions. There is minimal flow control. For flexibility, it is explicitly left to higher level protocols to provide these functions.

3. Main Features

The following paragraphs describe in some detail the mechanisms of the IP. A summary of the contents of the IP header is shown in Figure 3. Further information may be found in the current specification [1].

3.1. Addressing

The IP provides a two level addressing hierarchy. The upper level of the hierarchy is the *network number* (8 bits), and the lower level is an address within that network (24 bits), and is commonly called the *host*. This second level of the hierarchical address is sometimes called the *local address*. The details of the local address are dependent on the particular network.

The local address should allow a single physical host to act as several logically distinct internet hosts. That is, there should be mapping between internet

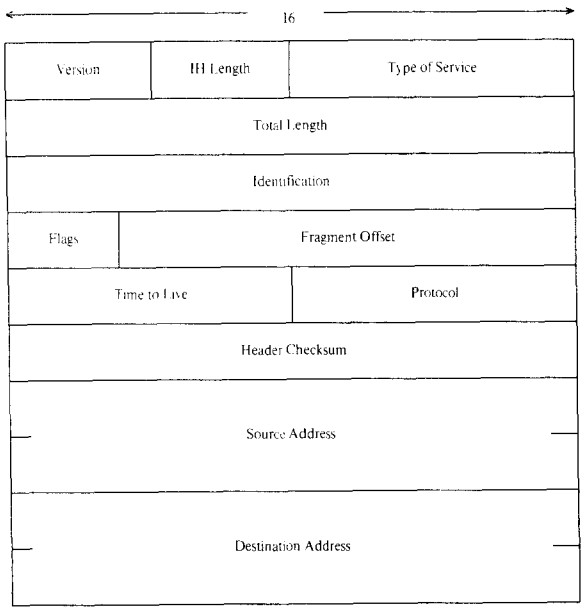


Fig. 3. INTERNET Protocol Header.

host addresses and network/host interfaces that allows several internet addresses to correspond to one physical interface. It should also be possible for several interfaces to accept or emit datagrams for the same internet address.

3.2. Protocol Number

The *Protocol Number* indicates the next level protocol used in the data portion of the datagram. This allows the internet module to demultiplex the incoming datagrams to higher level protocol modules for further processing. Hence, the protocol number indicates the format for parsing the rest of the datagram. Note that there is only one protocol number rather than a source protocol and a destination protocol because, higher level protocol modules exchange datagrams with each other using the same protocol. For example, two TCP modules exchange TCP segments via datagrams marked "TCP" in the protocol number.

One particular protocol number designates a multiplexing protocol which allows several independent data blocks from possibly different higher level protocol modules to be aggregrated together into one datagram for transmission [13].

3.3. Fragmentation and Reassembly

The IP provides information to allow datagrams to be fragmented for passage through networks with small packet size limits and to be reassembled at the destination. The necessary information includes an identification of the fragments that belong to the same datagram and the position of each fragment within the datagram.

The *Identification* (ID) field is used together with the source and destination address, and the protocol number, to identify datagram fragments to be assembled together. The *More Fragments* flag (MF) is set if the datagram is not the last fragment. The *Fragment Offset* (FO) identifies the fragment location, relative to the beginning of the original unfragmented datagram. These offsets are counted in units of 8 octets. Hence, if a datagram is fragmented, its data portion must be broken on 8 octet boundaries. This convention is designed so than an unfragmented datagram has all zero fragmentation information (MF = 0, FO = 0).

If the *Don't Fragment* flag (DF) is set, then internet fragmentation of this datagram is not permitted, although this may force it to be discarded at a gateway to a small packet network. DF can be used to prohibit fragmentation in cases where the receiving host does not wish to reassemble internet fragments. It is also possible that a small packet network could use network specific fragmentation and reassembly without the knowledge or involvement of the IP modules [14].

If a datagram is too large to be forwarded through any net, the entrance gateway breaks it into as many fragments as are necessary to fit within that net's packet size limit. Figure 4 shows a large datagram of 452 octets being fragmented into two smaller fragments (only the header fields relevant to fragmentation are given). Subsequent gateways may break the fragments into even smaller fragments if necessary using the same procedure.

Datagrams arriving at the destination IP are easily recognizable as fragments if either MF or FO is nonzero. Fragments from the same original datagram are identified by having identical ID fields (for a particular source, destination, and protocol number). Fragments are queued until the original datagram can be fully reassembled. Reassembly may be accomplished by placing the data from each fragment in a buffer at the position indicated by FO. Using the header information from the first fragment, the reas-

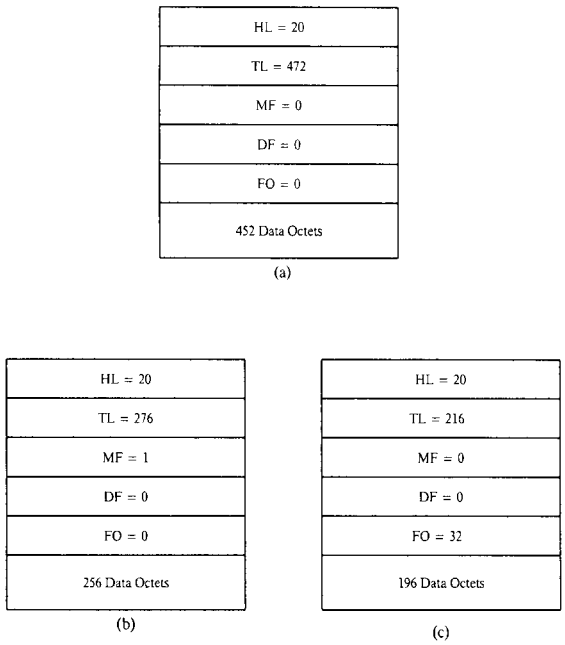


Fig. 4. Fragmentation Example.

sembled datagram is processed further just as if it had been received intact. If the time to live on any fragment expires during reassembly, the partially assembled datagram is discarded, and an error datagram is sent to the source.

A convention has been established in the current ARPA-Catenet that no datagrams larger that 576 octets will be sent, and that all receivers will be prepared to receive a reassemble datagrams up to this length (unless specifically arranged otherwise). This number is chosen to allow a data block of 512 octets and a reasonable number of header octets for several protocol levels to be transmitted in one datagram. Note that the IP header is repeated in each fragment. Hence, the minimum maximum packet size for any network in the Catenet is 20 header octets plus 8 data octets or 28 octets total.

The internet fragmentation procedure allows the fragments to be treated as independent datagrams the rest of the way to their destination (even taking different routes), with reassembly occurring only at the destination.

There is a need to uniquely identify the fragments of a particular datagram. Hence the sender must choose the identification field to be unique for each source/destination pair and protocol number for the time the datagram (or any fragment of it) could exist in the internet. Since the ID field allows 65,536 different values, some host may be able to simply use unique identifiers independent of destination.

It is beneficial for some higher level protocols to choose the identification field. For example, TCP protocol modules may retransmit an identical TCP segment, and the probability for correct reception would be enhanced if the retransmission carried the same identifier as the original transmission since fragments of either datagram could be used to construct a correct TCP segment. Note that a retransmission might be routed via a different set of networks and gateways and also may be fragmented into a different number of different sized fragments. The fragmentation information permits reassembly from fragments from either copy of the datagram.

3.4. Type of Service

The Type of Service (TOS) provides a network independent indication of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network. Some networks offer several precedence levels of service. Another choice involves a low-delay vs. high-reliability trade off. Typically networks invoke more complex (and delay producing) mechanisms as the need for reliability increases. A few networks offer a stream service, whereby one can achieve a "smoother" service at some cost. Typically this involves the reservation of resources within the network.

The abstract service quality parameters provided by IP are:

Precedence: Indicates the importance of this datagram.

Stream or Datagram: Indicates if there will be other datagrams from this source to this destination at regular frequent intervals justifying the maintenance of stream processing information.

Reliability: A measure of the level of effort desired to ensure delivery of this datagram.

Speed: A measure of the importance of prompt delivery of this datagram.

Speed over Reliability: Indicates the relative importance of speed and reliability when a conflict arises in achieving both.

3.5. Time to Live

The Time to Live (TTL) indicates the maximum time the datagram is allowed to exist in the Catenet.

As a datagram moves through the Catenet the TTL is decremented. If the TTL reaches zero the datagram should be discarded. The intention is to cause long delayed or undeliverable datagrams to be discarded. Guaranteeing a maximum lifetime for datagrams is important for the correct functioning of some higher level protocols such as TCP, and to protect the Catenet resources.

This field should be decreased at each point that the internet header is processed to reflect the time spent processing the datagram. Even if no information is available on the time actually spent, the field should be decremented by 1. The time is measured in units of seconds, and the maximum TTL is 255 seconds.

3.6. Checksum

The IP provides a checksum on the header only. Since some header fields may change (e.g., TTL, MF, FO), this is recomputed and verified at each point that the internet header is processed. This is a hop-by-hop checksum.

This checksum at the internet level is intended to protect the internet header fields from transmission errors. If the internet header contained undetected errors, misrouting and other unanticipated behavior could result. There may be applications in which it is desirable to receive data even though there are a few bit errors. If the IP enforced a data checksum and discarded datagrams with data checksum failures such applications would be restricted unnecessarily.

The checksum is computed as the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero. This checksum is simple to compute and has been adequately reliable for usage to date, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

3.7. Header Format

In addition to the main features discussed above, the IP includes the following items in the datagram header:

A *Version Number* (VER) which indicates the version of the IP in use, and hence the format of the internet header.

The *Internet Header Length* (IHL) is the length of the internet header and thus points to the beginning of the data.

The *Total Length* (TL) is the length of the datagram, including internet header and data. There are several protocol options, some of which are discussed in the next section.

4. Additional Features

The following optional mechanisms are available in the IP for use when needed.

4.1. Source Routing

The *Source Route* option provides a means for the source of a datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination.

As described above, routing at each gateway is based on the internet address in the destination field of the datagram header. If the source routing option is used, a series of additional internet addresses will be present in the option field. When the address in the destination field has been reached and the source route is not empty, the next address from the source route becomes the new destination (and is deleted from the source route list).

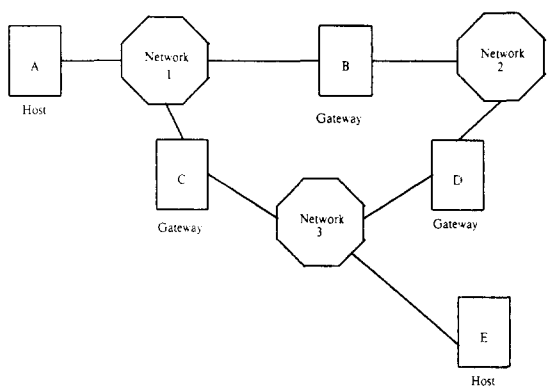


Fig. 5. Source Routing Example.

Thus, the source specifies a series of points the datagram must pass through on the way to its final destination. Normal internet routing is used to reach each of these points in turn, and the datagram may pass through a number of intermediate points between the specified addresses. Source routing may be used to specify routes to networks that are not known to the full internet system.

In Figure 5 an example of source routing is shown. Here host A is sending a datagram to host E. The normal routing would most likely be through the gateway C. We assume the user at host A would prefer in this case to have this datagram routed through gateways B and D. The Figure shows the address information at each step along the route.

4.2. Return (or Record) Route

The *Return Route* option provides a means to record the route taken by a datagram. A return route is composed of a series of internet addresses. When an IP module routes a datagram and the return route option is present, the gateway inserts its own internet address (in the environment of the next destination) into the return route option data.

4.3. Error Report

The *Error Report* option is used to report an error detected in processing a datagram to the source. A code indicates the type of error detected, and the ID is copied from the datagram in error, and additional octets of error information may be present depending on the error code. If a datagram consisting only of an error report option is found to be in error or must be discarded, no error report is sent.

Error codes are defined to report the following conditions: (0) No reason given, (1) Not Accepted – no program at the destination will accept the datagram, (2) Fragmentation Problem – the datagram cannot be delivered without fragmenting and the DF flag is set, (3) Reassembly Problem – the datagram cannot be reassembled because there are missing fragments and the time to live has expired, and (4) Gateway Congestion – the datagram was discarded to relieve congestion.

5. Gateway Functions

This section summarizes the tasks performed by a gateway; which are, interfacing to the local networks,

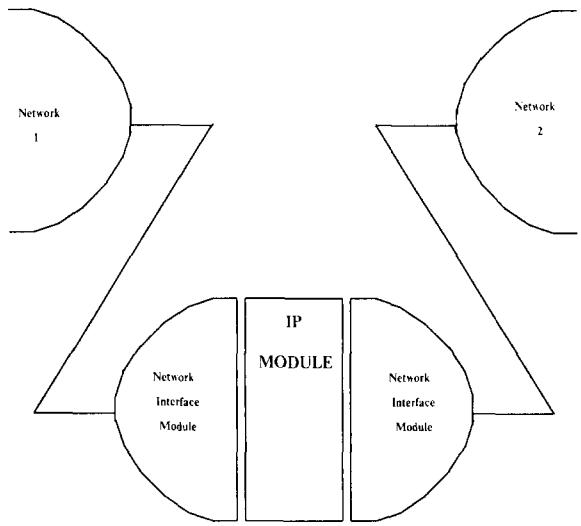


Fig. 6. Gateway.

and performing the IP functions.

The actual interconnection of networks is performed by gateways which are computers connected as hosts on several networks (see Figure 6). Messages are communicated across networks by using the protocols and conventions of the individual networks. While traversing each network the IP datagram is encapsulated within the local network protocols. At the gateway the IP datagram is decapsulated and examined by the gateway to determine how to route this datagram, and what local network options to use, if any. The gateway handles issues of routing, fragmentation (if the local network cannot handle regular size datagrams), error reporting and control, and interfacing to local networks.

The essential purpose of a gateway is to forward each datagram toward its destination. The key decision a gateway must make is the routing decision. When a gateway receives a datagram it must use the destination address in the IP header along with routing information stored in the gateway to determine where to send the datagram.

The routing information stored in the gateway may be relatively static (changed only by manual intervention) or dynamic (changed automatically). Both cases are allowed in the ARPA-Catenet system. The discussion of the techniques for dynamically updating the routing information are described by Strazisar [15].

Another important task of a gateway is to encapsulate datagrams for transmission through the next network, using that network's existing message trans-

fer protocol. This involves adding an appropriate message header (and perhaps trailer), to the datagram. The gateway must interpret the type of service field of the IP header to select the appropriate service in the next network.

The gateway decreases the TTL to account for the time elapsed since the TTL was last adjusted. This is an estimate of the time spent in transmission and processing. If this reduces the TTL to zero the gateway discards the datagram.

If the datagram is larger than the maximum packet size of the next network, the gateway may fragment it into pieces that will be sent separately.

If the gateway must discard a datagram due to congestion or errors in processing the datagram (such as an unknown or currently unreachable address), it sends an error report datagram to the source of the discarded datagram.

Of course, the gateway verifies the IP header checksum on every datagram it receives before processing it. If the check fails the datagram is discarded with no notification to the source or adjacent gateway. Since some of the IP header information is changed during gateway processing (e.g. TTL), the gateway computes a new IP header checksum before sending it on.

Each datagram can be processed completely independently of other datagrams. The provision of error recovery, sequencing, or flow control functions are left for end-to-end protocols, and the gateway does not maintain any status information or dedicate any resources for individual virtual circuits. Indeed, the gateway is unaware of any details of the higher protocol levels.

6. Design Decisions

The key decision in the design of the ARPA Internet Protocol is the choice of a datagram basis rather than a virtual circuit basis. Using datagrams as the basis of communication in the Catenet permits the use of simpler gateways since they are not required to maintain state information about the individual virtual circuits, and allows the end-to-end communication to continue via alternate routing if a gateway fails.

Using datagrams as the basic communication service allows the construction of virtual circuit style end-to-end services (e.g., TCP), and other services. In the DARPA research program there are needs for other styles of communication service. For example, the packet speech requires a service which provides minimal delay even at the cost of a few dropped messages. Such a service can be built on a datagram base, but not on a virtual circuit base. For more detail on the tradeoff between a datagram base and a virtual circuit base for communications see references [8–12].

This choice of a datagram base for the operation of the Catenet results in the separation of the internet protocol from the end-to-end protocols in general and TCP in particular. The early proposals for TCP did not focus clearly on the responsibilities of the gateways and did not allow for alternate styles of communication service. Once these needs were apparent the protocol functions were separated into distinct layers.

The decision to use the encapsulation/decapsulation technique to send the IP datagrams through local nets was made to maximize individual networks' autonomy, and to avoid the need for modifications of individual networks (particularly in the area of routing) to support internet traffic [10].

The decision to fragment datagrams in gateways as they pass from a large packet network into a small packet network, but not reassemble the fragments until they reach the destination host, allows simpler gateways and minimizes the delay in the Catenet. The alternate approach of reassembly in the next gateway is explored in reference [14].

Perhaps the most difficult design decision was the choice of the address size and structure. The size of the address field is a compromise that allows enough addresses for the anticipated growth of the Catenet yet is not an excessive overhead burden. The structuring of the address into network and host fields allows the gateways to process datagrams destined for distant networks on the basis of just the network field. This field separation also reflects an administrative delegation of the address assignment function.

In addition to the address, IP carries additional address or multiplexing information in the protocol field. This indicates which next level protocol should be used to interpret this datagram. Most of the higher level protocols have further multiplexing information called *ports* in their headers. The IP approach to addressing may be characterized as hierarchical [10].

An option in IP supports the concept of source routing. This means a source may specify a series of addresses which are used in turn until the ultimate destination is reached [10]. The decision to include

this feature was motivated by the realization that many small networks may be interconnected to the Catenet via ad hoc arrangements, and destinations in such networks (or such networks themselves) may be unknown to gateways in the general Catenet.

IP uses a Time To Live which is decremented by each gateway by at least one unit (more if the datagram is delayed in the gateway for a substantial time). Other protocols use a hop count which is incremented by each gateway [5]. The practical difference is small, though the time to live approach remains effective as the size of the network changes, and allows the source to specify a maximum life for the datagram.

7. Research Issues

7.1. Multiple Addresses

There are several issues related to more flexible addressing that the current IP does not deal with. One case is a host with two (or more) internet addresses, either on one network or even on different networks. Sometimes this serves to distinguish between logically separate hosts, but in other cases it is desirable to consider both addresses as the "same place" as far as higher level protocols are concerned. It is not clear how a gateway could know when or how to route messages sent to one address to another address (e.g. if the first address was unreachable). A particularly difficult example of this problem is a mobile packet radio which moves from one network to another while trying to maintain unbroken communication.

7.2. Local Networks

A second issue is the addressing of local networks. There will soon be a large number of local networks (e.g., networks within one building or on a campus) wishing to use the ARPA-Catenet for long distance interconnection. It seems unreasonable that every one of these should have the same status as a nationwide network, with all gateways responsible for maintaining routing information about them. It may be preferable to introduce another level in the addressing hierarchy, or to combine a gateway plus internal address for such nets in the local address field of IP addresses [16].

7.3. Multiple Destinations

Another addressing issue is provision of a capability to send datagrams to a number of destinations at once. Broadcast to all is, of course, the ultimate multi-destination, but "to all" is easier to handle then "to some." This capability is inherent in the technology of some networks (e.g. satellite, ring, and Ethernets) but there is no provision in the current IP for such multidestination addressing. These is work underway in the ARPA community on an internetwork digital packet speech conferencing experiment. A protocol called ST developed for that experiment does contain a multidestination capability [17].

7.4. Naming/Addressing/Routing

The mapping of character string names that are convenient for people into internet addresses is often a problem. This can be eased by the provision of a "directory assistance" service or name server [18]. A name server is a service with a table of name/address correspondences. When the name server is sent a query about a name it responds with the name and corresponding address(es). Directory services can be provided in a centralized and/or distributed fashion. For a further discussion of the roles of names, addresses, and routes see [19].

7.5. Congestion Control

Congestion control is a problem for any network. The gateways may be viewed as nodes of the Catenet, much as IMPs are the nodes of the ARPANET. As internet traffic increases, gateways may become overloaded, even while the individual networks connecting them are enforcing their own congestion controls. Thus there may be a need for an internet congestion control mechanism which is effective with the datagram mode of operation in the Catenet. Several methods such as isarithmic control, buffer categories, and "choke" packets [20] have been proposed for such environments. The ARPA gateways implement a simple strategy of notifying the source when a packet must be discarded due to congestion.

7.6. Monitoring and Adminstrative Control

Accounting is another basic internetworking requirement. Traffic statistics are useful for monitoring and control purposes, and are easily collected by

the gateways either on a net-to-net basis, or with more detail by internet source/destination pairs. Volume of packets and/or bits can be collected by a set of counters, and periodically dumped to a Catenet monitoring and accounting center. A gateway monitoring and control center is now operating to coordinate the collection of these statistics [21].

8. Conclusions

The ARPA Internet Protocol provides a common base for supporting higher level protocols in a network independent multi-network environment. The datagram basis of the internet protocol has allowed the flexible evolution of a variety of application specific higher level protocols while allowing simple gateways to interconnect networks. The principle of encapsulation for transmission through individual networks is essential for the provision of internet service over a variety of networks without requiring changes to each networks' internal operation.

As of August 1980, IP is implemented in 12 gateways interconnecting 10 networks, including packet radio, satellite, local nets, and the original ARPANET. Gateways are typically PDP 11/40 or 11/03 processors with limited memory. High level protocols including TCP, terminal access (Telnet), and file transfer (FTP) are in use above IP. Transaction oriented services such as directory assistance (Name Server) are also in use. Other applications are under development.

Acknowledgments

IP has developed in the context of a multicontractor research effort. It is not possible to list all the contributors that have participated in the DARPA Internet Program, however special mention should be made of the coordinating effort and technical contributions of Dr. Vinton G. Cerf of DARPA.

References

[1] Postel, J., "DOD Standard Internet Protocol," IEN 128, RFC 760, USC/Information Sciences Institute, NTIS document number ADA079730, January 1980.

[2] Cerf, V., "The Catenet Model for Internetworking," IEN 48, Defense Advanced Research Projects Agency, July 1978.

[3] Postel, J., "DOD Standard Transmission Control Protocol," IEN 129, RFC 761, USC/Information Sciences Institute, NTIS document number ADA082609, January 1980.

[4] Cerf, V. and R. Kahn, "A Protocol for Packet Network Intercommunication," IEEE Transactions on Communications, vol. COM-22, no. 5, May 1974.

[5] Boggs, D.R., et al., 'Pup: An Internetwork Architecture," IEEE Transactions on Communications, vol. COM-28, no. 4, April 1980.

[6] Shoch, J., D. Cohen, and E. Taft, "Mutual Encapsulation of Internet Protocols," Trends and Applications 1980: Computer Network Protocols, National Bureau of Standards, Gaithersbug, Maryland, May 1980.

[7] Deparis, M., et. al., "The Implementation of an End-to-End Protocol by EIN Centers: A Survey and Comparison," Proceedings International Conference on Computer Communication, Toronto, Canada, August 1976.

[8] Grossman, G.R., A. Hinchley, and C.A. Sunshine, "Issues in International Public Data Networking," Computer Networks, vol. 3, no. 4, August 1979.

[9] DiCiccio, V., et. al., "Alternatives for Interconnection of Public Packet Switching Networks," Proceedings. Sixth Data Communication Symposium, ACM/IEEE, November 1979.

[10] Sunshine, C., "Interconnection of Computer Networks," Computer Networks vol. 1, no. 3, pp. 175–195, January 1977.

[11] Cerf, V. and P. Kirstein, "Issues in Packet-Network Interconnection," Proceedings of the IEEE, vol. 66, no. 11, November 1978.

[12] Postel, J., "Internetwork Protocol Approaches," IEEE Transactions on Communications, vol. COM-28, no. 4, April 1980.

[13] Cohen, D. and J. Postel, "On Protocol Multiplexing," Sixth Data Communication Symposium, ACM/IEEE, November 1979.

[14] Shoch, J., "Packet Fragmentation in Internetwork Protocols," Computer Networks, vol. 3, no. 1, February 1979.

[15] Strazisar, V., "How to Build a Gateway," IEN 109, Bolt, Beranek, and Newman, August 1979.

[16] Cohen, D., "On Addressing and Related Issues (Or Fuel for a Discussion)," IEN 122, USC/Information Sciences Institute, October 1979.

[17] Forgie, J., "ST – A Proposed Internet Stream Protocol," IEN 119, M.I.T. Lincoln Laboratory, September 1979.

[18] Pickens, J., E. Feinler, and J. Mathis, "The NIC Name Server – A Datagram Based Information Utility," Proceedings of the Fourth Berkeley Conference on Distributed Data Management and Computer Networks, August 1979.

[19] Shoch, J., "Inter-Network Naming, Addressing, and Routing," COMPCON Fall 78 Proceedings, (IEEE Catalog Number 78CH-1388-8C), Washington, D.C., September 1978.

[20] Grange, J., and M. Gien, eds., Flow Control in Computer Networks, North Holland, February 1979.

[21] Flood Page, D., "ARPA Catenet Monitoring and Control," IEN 105, Bolt, Beranek, and Newman, May 1979.