TCP/IP Overview

The generic term "TCP/IP" usually means anything and everything related to the specific protocols of TCP and IP. It can include other protocols, applications and even the network medium. A sample of these protocols are: UDP, ARP, and ICMP. A sample of these applications are: TELNET, FTP, and RCP. A more accurate term is "internet technology". A network that uses internet technology is called an "internet".

Internet Protocols

TCP is a datagram, or connectionless, internetwork service and includes provision for addressing, type-of-service specification, fragmentation and reassembly, and security information. ICMP is a connectionless routing service provided by the User Datagram Protocol (UDP).

Reliable data delivery is provided in the internet protocol suite by transport-level protocols such as the Transmission Control Protocol (TCP), which provides end-to-end retransmission, resequencing and connection control. Transport-level connectionless service is provided by the User Datagram Protocol (UDP).

Elements of the Internet System

The internet environment consists of hosts connected to networks which in turn are interconnected via gateways. It is assumed here that the networks may be either local networks (e.g., the Ethernet) or large networks (e.g., the ARPANET), but in any case are based on packet switching technology. The active agents that produce and consume messages are processes. Various levels of protocols in the networks, the gateways, and the hosts support an interprocess communication system that provides two-way data flow on logical connections between process ports.

The constituent networks of the Internet system are required only to provide packet (connectionless) transport. This requires only delivery of individual packets. According to the IP service specification, datagrams can be delivered out of order, be lost or duplicated and/or contain errors. Reasonable performance of the protocols that use IP (e.g., TCP) requires an IP datagram loss rate of less than 5%.

The term packet is used generically here to mean the data of one transaction between a host and its network. The format of data blocks exchanged within the a network will generally not be of concern to us. Hosts are computers, and from the communication network's point of view, are the sources and destinations of packets. Processes are viewed as the active elements in host computers (in accordance with the fairly common definition of a process as a program in execution). Even terminals and files or other I/O devices are viewed as communicating with each other through the use of processes. Thus, all communication is viewed as inter-process communication.

Since a process may need to distinguish among several communication streams between itself and another process (or processes), we imagine that each process may have a number of ports through which it communicates with the ports of other processes.

Flow of Data

Let's follow the data as it flows down through the protocol stack shown below. For an application that uses TCP (Transmission Control Protocol), data passes between the application and the TCP module. For applications that use UDP (User Datagram Protocol), data passes between the application and the UDP module. FTP (File Transfer Protocol) is a typical application that uses TCP. Its protocol stack in this example is FTP/TCP/IP/ENET. SNMP (Simple Network Management Protocol) is an application that uses UDP. Its protocol stack in this example is SNMP/UDP/IP/ENET.

The TCP module, UDP module, and the Ethernet driver are n-to-1 multiplexers. As multiplexers they switch many inputs to one output. They are also 1-to-n demultiplexers. As demultiplexers they switch one input to many outputs according to the type field in the protocol header.

If an Ethernet frame comes up into the Ethernet driver off the network, the packet can be passed upwards to either the ARP (Address Resolution Protocol) module or to the IP (Internet Protocol) module. The value of the type field in the Ethernet frame determines whether the Ethernet frame is passed to the ARP or the IP module.

If an IP packet comes up into IP, the unit of data is passed upwards to either TCP or UDP, as determined by the value of the protocol field in the IP header.

If the UDP datagram comes up into UDP, the application message is passed upwards to the network application based on the value of the port field in the UDP header. If the TCP message comes up into TCP, the application message is passed upwards to the network application based on the value of the port field in the TCP header.

The downswards multiplexing is simple to perform because from each starting point there is only the one downward path; each protocol module adds its header information so the packet can be demultiplexed at the destination computer.

Data passing out from the applications through either TCP or UDP converges on the IP module and is sent downwards through the lower network interface driver.

This is the logical structure of the layered protocols inside a computer on an internet. Each computer that can communicate using internet technology has such a logical structure. It is this logical structure that determines the behavior of the computer on the internet. The boxes represent processing of the data as it passes through the computer, and the lines connecting boxes show the data path. The horizontal line at the bottom represents the Ethernet cable; the "o" is the transceiver. The "a" is the IP address and the "g" is the Ethernet address.
IP Addressing

Addresses are fixed length of four octets (32 bits). An address begins with a network number, followed by local address (called the "rest" field).

The network manager assigns IP addresses to computers according the IP network to which the computer is attached. One part of a 4-byte IP address is the IP network number, the other part is the IP computer number (or host number).

The IP address space is administered by the NIC (Network Information Center). All internets that are connected to the single world-wide Internet must use network numbers assigned by the NIC. A distinction is made between names, addresses, and routes. A name indicates what we seek. An address indicates where it is. A route indicates how to get there. The internet protocol deals primarily with addresses. It is the task of higher level (i.e., host-to-host or application) protocols to use the addressing to map names to addresses. The internet module maps internet addresses to local net addresses. It is the task of lower level (i.e., local net or gateways) procedures to make the mapping from local net addresses to routes.

Care must be taken in mapping internet addresses to local net addresses; a single physical host must be able to act as if it were several distinct hosts to the extent of using several logical internet addresses. That is, provision must be made for a host to have several physical interfaces to the network with each having several logical internet addresses.

Subnets

An IP datagram carries 32-bit source and destination addresses, each of which is partitioned into two parts -- a constituent network number and a host number on that network. Symbolically:

IP-address ::= {<Network-number>, <Host-number>}

To finally deliver the datagram, the last gateway in its path must map the host-number (or "rest") part of an IP address into the physical address of a host connection to the constituent network.

This simple notion has been extended by the concept of "subnets", which were introduced in order to allow arbitrary complexity of interconnected LAN structures within an organization, while insulating the Internet system against explosive growth in network numbers and routing complexity. Subnets essentially provide a two-level hierarchical routing structure for the Internet system. The basic idea is to partition the <host number> field into two parts: a subnet number, and a true host number on that subnet.

IP-address ::= {<Network-number>, <Subnet-number>, <Host-number>}

The interconnected LANs of an organization will be given the same network number but different subnet numbers. The distinction between the subnets of such a subnetted network must not be visible outside that network. Thus, wide-area routing in the rest of the Internet will be based only upon the <Network-number> part of the IP destination address; gateways outside the network will lump <Subnet-number> and <Host-number> together to form an uninterpreted "rest" part of the 32-bit IP address. Within the subnetted network, the local gateways must route on the basis of an extended network number:

{<Network-number>, <Subnet-number>}

Flexible use of the available address space will be increasingly important in coping with the anticipated growth of the Internet. Thus, we allow a particular subnetted network to use more than one subnet mask. Several campuses with very large LAN configurations are also creating nested hierarchies of subnets, sub-subnets, etc.

Internet Protocol (IP) Operation

The internet protocol implements two basic functions: addressing and fragmentation.

The internet modules use the functions called in the internet header to transmit internet datagrams toward their destinations. The selection of a path for transmission is called routing.

The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol does not provide a reliable communication facility. There are no acknowledgments 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 no flow control.

Errors detected may be reported via the Internet Control Message Protocol (ICMP) which is implemented in the internet protocol module.

IP Routing

The model of operation for transmitting a datagram from one application program to another is illustrated by the following scenario:

We suppose that this transmission will involve one intermediate gateway.

The sending application program prepares its data and calls on its local internet module to send that data as a datagram and passes the destination address and other parameters as arguments of the call.

The internet module prepares a datagram header and attaches the data to it. The internet module determines a local network address for this internet address, in this case it is the address of a gateway.

It sends this datagram and the local network address to the local network interface.

The local network interface creates a local network header, and attaches the datagram to it, then sends the result via the local network.

The datagram arrives at a gateway host wrapped in the local network header, the local network interface strips off this header, and turns the datagram over to the internet module. The internet module determines from the internet address that the datagram is to be forwarded to another host in a second network. The internet module determines a local net address for the destination host. It calls on the local network interface for that network to send the datagram.

This local network interface creates a local network header and attaches the datagram sending the result to the destination host.

At this destination host the datagram is stripped of the local net header by the local network interface and handed to the internet module.

The internet module determines that the datagram is for an application program in this host. It passes the data to the application program in response to a system call, passing the source address and other parameters as results of the call.

Internet Header Format

A summary of the contents of the internet header follows:

Example Internet Datagram Header

Transmission Control Protocol

TCP is used by network applications that require guaranteed delivery and cannot be bothered with doing time-outs and retransmissions. The two most typical network applications that use TCP are File Transfer Protocol (FTP) and the TELNET. Other popular TCP network applications include X-Windows System, rcp (remote copy), and the r-commands. TCP's greater capability is not without cost: it requires more CPU and network bandwidth. The internals of the TCP module are much more complicated than those in a UDP module.

Similar to UDP, network applications connect to TCP ports. Well-defined port numbers are dedicated to specific applications. For instance, the TELNET server uses port number 23. The TELNET client can find the server simply by connecting to port 23 of TCP on the specified computer.
When the application first starts using TCP, the TCP module on the client’s computer and the TCP module on the server’s computer start communicating with each other. These two end-point TCP modules contain state information that defines a virtual circuit. This virtual circuit consumes resources in both TCP end-points. The virtual circuit is full duplex; data can go in both directions simultaneously. The application writes data to the TCP port, the data traverses the network and is read by the application at the far end.

TCP packetizes the byte stream at will; it does not retain the boundaries between writes. For example, if an application does 5 writes to the TCP port, the application at the far end might do 10 reads to get all the data. Or, if the local host gets all the data with a single read. There is no correlation between the number and size of writes at one end to the number and size of reads at the other end.

TCP is a sliding window protocol with time-out and retransmits. Outgoing data must be acknowledged by the far-end TCP. Acknowledgments can be piggybacked on data. Both receiving ends can flow control the far end, thus preventing a buffer overrun.

As with all sliding window protocols, the protocol has a window size. The window size determines the amount of data that can be transmitted before an acknowledgment is required. For TCP, this amount is not a number of TCP segments but a number of bytes.

The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in hosts connected to distinct but interconnected computer communication networks. Very few assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols. In principle, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks.

TCP Operation

As noted above, the primary purpose of the TCP is to provide reliable, secure logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable internet communication system requires facilities in the following areas:

- Basic Data Transfer
- Reliability
- Flow Control
- Multiplexing
- Connections (precedence and security)

Reliability:
The TCP must recover from data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by inserting a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may have been received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.

Flow Control:
TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a “window” with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.

Multiplexing:
To allow for many processes within a single Host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenated with the network and host addresses from the internet communication layer, this forms a socket. A pair of sockets uniquely identifies each connection. That is, a socket may be simultaneously used in multiple connections.

The binding of ports to processes is handled independently by each Host. However, it proves useful to attach frequently used processes (e.g., a “logger” or timesharing service) to fixed sockets which are made known to the public. These services can then be accessed through the known addresses. Establishing and learning the port addresses of other processes may involve more dynamic mechanisms.

Connections:
The reliability and flow control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is uniquely specified by a pair of sockets identifying its two sides.

When two processes wish to communicate, their TCP’s must first establish a connection (initialize the status information on each side). When their communication is complete, the connection is terminated or closed to free the resources for other uses.

Since connections must be established between unreliable hosts and over the unreliable internet communication system, a handshake mechanism with clock-based sequence numbers is used to avoid erroneous initialization of connections.

Model of Operation

Processes transmit data by calling on the TCP and passing buffers of data as arguments. The TCP packages the data from these buffers into segments and calls on the internet module to transmit each segment to the destination TCP. The receiving TCP places the data from a segment into the receiving user’s buffer and notifies the receiving user. The TCPs include control information in the segments which they use to ensure reliable ordered data transmission.

The model of internet communication is that there is an internet protocol module associated with each TCP which provides an interface to the local network. This internet module packages TCP segments inside internet datagrams and routes these datagrams to a destination internet module or intermediate gateway. To transmit the datagram through the local network, it is embedded in a local network packet.

The packet switches may perform further packaging, fragmentation, or other operations to achieve the delivery of the local packet to the destination internet module.

At a gateway between networks, the internet datagram is “unwrapped” from its local packet and examined to determine through which network the internet datagram should travel next. The internet datagram is then “wrapped” in a local packet suitable to the next network and routed to the next gateway, or to the final destination.

A gateway is permitted to break up an internet datagram into smaller internet datagram fragments if this is necessary for transmission through the next network. To do this, the gateway produces a set of internet datagrams; each carrying a fragment. Fragments may be further broken into smaller fragments at subsequent gateways. The internet datagram fragment format is designed so that the destination internet module can reassemble fragments into internet datagrams.

A destination internet module unwraps the segment from the datagram (after reassembling the datagram, if necessary) and passes it to the destination TCP.

This simple model of the operation glosses over many details. One important feature is the type of service. This provides information to the gateway (or internet module) to guide it in selecting the service parameters to be used in traversing the next network. Included in the type of service information is the precedence of the datagram. Datagrams may also carry security information to permit host and gateways that operate in multillevel secure environments to properly segregate datagrams for security considerations.

The Host Environment

The TCP is assumed to be a module in an operating system. The users access the TCP much like they would access the file system. The TCP may call on other operating system functions, for example, to manage data structures. The actual interface to the network is assumed to be controlled by a device driver module. The TCP does not call on the network device driver directly, but rather calls on the internet datagram protocol module which may in turn call on the device driver.

The mechanisms of TCP do not preclude implementation of the TCP in a front-end processor. However, in such an implementation, a host-to-front-end protocol must provide the functionality to support the type of TCP-user interface described in this document.

TCP Interfaces

The TCP/user interface provides for calls made by the user on the TCP to OPEN or CLOSE a connection, to SEND or RECEIVE data, or to maintain STATUS about a connection. These calls are like other calls from user programs on the operating system, for example, the calls to open, read from, and close a file.

The TCP/internet interface provides calls to send and receive datagrams addressed to TCP modules in hosts anywhere in the internet sys-
strategy would have the local connection name be a pointer to the TCB for this connection. The OPEN call also specifies whether the connection establishment is to be actively pursued, or to be passively waited for.

Well-known sockets are a convenient mechanism for a priori associating a socket address with a standard service. For instance, the “Telnet-Server” process is permanently assigned to a particular socket, and other sockets are reserved for File Transfer, Remote Job Entry, Text Generator, Echoer, and Sink processes (the last three being for test purposes). A socket address might be reserved for access to a “Look-Up” service which would return the specific socket at which a newly created service would be provided. The concept of a well-known socket is part of the TCP specification, but the assignment of sockets to services is outside the scope of this discussion. Processes can issue passive OPENs and wait for matching active OPENs from other processes and be informed by the TCP when connections have been established. Two processes which issue active OPENs to each other at the same time will be correctly connected. This flexibility is critical for the support of distributed computing in which components act asynchronously with respect to each other.

Data Communication

The data that flows on a connection may be thought of as a stream of octets. The sending user indicates in each SEND call whether the data in that call (and any preceeding calls) should be immediately pushed through to the receiving user by the setting of the PUSH flag.

A sending TCP is allowed to collect data from the sending user and to send that data in segments at its own convenience, until the push function is signaled, then it must send all unsent data. When a receiving TCP sees the PUSH flag, it must not wait for more data from the sending TCP before passing the data to the receiving process.

There is no necessary relationship between push functions and segment boundaries. The data in any particular segment may be the result of a single SEND call, in whole or part, or of multiple SEND calls.

The purpose of push function and the PUSH flag is to push data through from the sending user to the receiving user. It does not provide a record service.

TCP segments are sent as internet datagrams. The Internet Protocol header carries several information fields, including the source and destination host addresses. A TCP header follows the internet header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>Sequence Number</th>
<th>Acknowledgment Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

TCP Header Format

(Note that one tick mark represents one bit position.)

TCP segments are sent as internet datagrams. The Internet Protocol header carries several information fields, including the source and destination host addresses. A TCP header follows the internet header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

Source Port: 16 bits
The source port number.
Destination Port: 16 bits
The destination port number.
Sequence Number: 32 bits
The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

TCP segments are sent as internet datagrams. The Internet Protocol header carries several information fields, including the source and destination host addresses. A TCP header follows the internet header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

Source Port: 16 bits
The source port number.
Destination Port: 16 bits
The destination port number.
Sequence Number: 32 bits
The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

TCP also provides a means to communicate to the receiver of data that at some point further along in the data stream than the receiver is currently reading there is urgent data. TCP does not attempt to define what the user specifically does upon being notified of pending urgent data, but the general notion is that the receiving process will take action to process the urgent data quickly.

Data Offset: 4 bits
The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.

Reserved: 6 bits
Reserved for future use. Must be zero.

Control Bits: 6 bits (from left to right):
URG: Urgent Pointer field significant ACK: Acknowledgment field significant PSH: Push Function RST: Reset the connection SYN: Synchronize sequence numbers FIN: No more data from sender

Window: 16 bits
The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to
Window Management Suggestions

Allocating a very small window causes data to be transmitted in many small segments when better performance is achieved using fewer large segments.

One suggestion for avoiding small windows is for the receiver to defer updating a window until the additional allocation is at least X percent of the maximum allocation possible for the connection (where X might be 20 to 40). Another suggestion is for the sender to avoid sending small segments by waiting until the window is large enough before sending data. If the the user signals a push function then the data must be sent even if it is a small segment.

Note that the acknowledgments should not be delayed or unnecessary retransmissions will result. One strategy would be to send an acknowledgment when a small segment arrives (with out updating the window information), and then to send another acknowledgment with new window information when the window is larger.

The segment sent to probe a zero window may also begin a breakup of transmitted data into smaller and smaller segments. If a segment containing a single octet data sent to probe a zero window is accepted, it consumes one octet of the window now available. If the sending TCP simply sends as much as it can whenever the window is non zero, the transmitted data will be broken into alternating big and small segments. As time goes on, occasional pauses in the receiver making window allocation available will result in breaking the big segments into a small and not quite so big pair. And after a while the data transmission will be in mostly small segments.

The suggestion here is that the TCP implementations need to actively attempt to combine small window allocations into larger windows, since the mechanisms for managing the window tend to lead to many small windows in the simplest minded implementations.

Physical and Link Layer

Multiaccess Functionality

This section is a short review of Ethernet technology.

An Ethernet frame contains the destination address, source address, type field, and data.

An Ethernet address is 6 bytes. Every device has its own Ethernet address and listens for Ethernet frames with that destination address. All devices also listen for Ethernet frames with a wildcard destination address of “FF-FF-FF-FF-FF-FF” (in hexadecimal), called a “broadcast” address.

Ethernet uses CSMA/CD (Carrier Sense and Multiple Access with Collision Detection). CSMA/CD means that all devices communicate on a single medium, that only one can transmit at a time, and that they can all receive simultaneously. If 2 devices try to transmit at the same instant, the transmit collision is detected, and both devices wait a random (but short) period before trying to transmit again.

A Human Analogy

A good analogy of Ethernet technology is a group of people talking in a small, completely dark room. In this analogy, the physical network medium is sound waves on air in the room instead of electrical signals on a coaxial cable.

Each person can hear the words when another is talking (Carrier Sense). Everyone in the room has equal capability to talk (Multiple Access), but none of them give lengthy speeches because they are polite. If a person is impolite, he is asked to leave the room (i.e., thrown off the net).

No one talks while another is speaking. But if two people start speaking at the same instant, each of them know this because such hands something they haven’t said (Collision Detection). When these two people notice this condition, they wait for a moment, then one begins talking. The other hears the waiting and waits for the first to finish before beginning his own speech.

Each person has an unique name (unique Ethernet address) to avoid confusion. Every time one of them talks, he prefaxes the message with the name of the person he is talking to and with his own name (Ethernet destination and source address, respectively), i.e., “Hello Jane, this is Jack...blah blah blah...”. If the sender wants to talk to everyone he might say “everyone” (broadcast address), i.e., “Hello Everyone, this is Jack...blah blah blah...”.

Physical and Link Layer

The goal of this specification is to allow compatible and interoperable implementations for transmitting IP datagrams and ARP requests and replies. To achieve this it may be necessary in a few cases to limit the use that IP and ARP make of the capabilities of a particular IEEE 802 standard.

The IEEE 802 specifications define a family of standards for Local
IEEE 802 packets may have a maximum size restriction. Implementations are encouraged to support full-length packets.

For compatibility purposes, the maximum packet size used with IP datagrams or ARP requests and replies must be consistent on a particular network.

Gateway implementations must be prepared to accept full-length packets and fragment them when necessary.

Host implementations should be prepared to accept full-length packets, however hosts must not send datagrams longer than 576 octets unless they have explicit knowledge that the destination is prepared to accept them. A host may communicate its size preference in TCP based applications via the TCP Maximum Segment Size option.

Datagrams on IEEE 802 networks may be longer than the general Internet default maximum packet size of 576 octets. Hosts connected to an IEEE 802 network should keep this in mind when communicating because sending smaller datagrams to avoid unnecessary fragmentation at intermediate gateways. Please see for further information.

IEEE 802.2 LLC and SNAP data link layers, and the 802.3, 802.4, or 802.5 physical networks layers. The SNAP is used with an Organization Code indicating that the following 16 bits specify the EtherType code (as listed in Assigned Numbers).

Header Format

<table>
<thead>
<tr>
<th>Protocol Id or Org Code</th>
<th>EtherType</th>
<th>SNAP</th>
</tr>
</thead>
</table>

Address Mappings

The hardware address length is 2 for 16-bit IEEE 802 addresses, or 6 for 48-bit IEEE 802 addresses.

IEEE 802 packets may have a minimum size restriction. Implementations are encouraged to support full-length packets.

For compatibility purposes, the maximum packet size used with IP datagrams or ARP requests and replies must be consistent on a particular network.

Gateway implementations must be prepared to accept full-length packets and fragment them when necessary.

Host implementations should be prepared to accept full-length packets, however hosts must not send datagrams longer than 576 octets unless they have explicit knowledge that the destination is prepared to accept them. A host may communicate its size preference in TCP based applications via the TCP Maximum Segment Size option.

Datagrams on IEEE 802 networks may be longer than the general Internet default maximum packet size of 576 octets. Hosts connected to an IEEE 802 network should keep this in mind when sending data-grams to hosts not on the same IEEE 802 network. It may be appropriate to send smaller datagrams to avoid unnecessary fragmentation at intermediate gateways. Please see for further information.

IEEE 802.2 Details:

While not necessary for supporting IP and ARP, all implementations are required to support IEEE 802.2 standard Class I service. This requires supporting Unnumbered Information (UI) Commands, eX-change IDentification (XID) Commands and Responses, and TEST link (TEST) Commands and Responses.

When either an XID or a TEST command is received a response must be returned; with the Destination and Source addresses, and the DSAP and SSAP swapped.

When responding to an XID or a TEST command the sense of the poll/final bit must be preserved. That is, a command received with the poll/final bit reset must have the response returned with the poll/final bit re-set and vice versa.

The XID command or response has an LLC control field value of 175 (decimal) if poll is off or 191 (decimal) if poll is on. (See Appendix on Numbers.)

The TEST command or response has an LLC control field value of 227 (decimal) if poll is off or 243 (decimal) if poll is on. (See Appendix on Numbers.)

A command frame is identified with high order bit of the SSAP address reset. Response frames have high order bit of the SSAP address set to one.

XID response frames should include an 802.2 XID Information field of 129.1.0 indicating Class I (connectionless) service. (type 1).

TEST response frames should echo the information field received in the corresponding TEST command frame.

For IEEE 802.3:

A particular implementation of an IEEE 802.3 Physical Layer is denot-ed using a three field notation. The three fields are data rate in megabit/second, medium type, and maximum segment length in hundreds of meters. One combination of of 802.3 parameters is 10BASE5 which
specifies a 10 megabit/second transmission rate, baseband medium, and 500 meter segments. This corresponds to the specifications of the familiar "Ethernet" network.

The MAC header contains 6 (2 octets of source address, 6 (2 octets of destination address, and 2 octets of length. The MAC trailer contains 4 octets of Frame Check Sequence (FCS), for a total of 18 (10) octets.

IEEE 802.3 networks have a minimum packet size that depends on the transmission rate. For type 10BASE5 802.3 networks the minimum packet size is 1518 octets.

IEEE 802.3 networks have a maximum packet size which depends on the transmission rate. For type 10BASE5 802.3 networks the maximum packet size is 1518 octets including all octets between the destination address and the FCS inclusive.

This allows: 1518 - 18 (MAC header+trailer) - 8 (LLC+SNAP header) = 1492 for the IP datagram (including the IP header). Note that 1492 is not equal to 1500 which is the MTU for Ethernet networks.

### Gateways

Gateways implement internet protocol to forward datagrams between networks. Gateways also implement the Gateway to Gateway Protocol (GGP) to coordinate routing and other internal control information.

In a gateway the higher level protocols need not be implemented and the GGP functions are added to the IP module.

| Internet Protocol & ICMP & GGP |
| Local Net | Local Net |

**Gateway Protocols**

In the Internet model, constituent networks are connected together by IP datagram forwards which are called "gateways" or "IP routers". In this document, every use of the term "gateway" is equivalent to "IP router". In current practice, gateways are normally realized with packet-switching software executing on a general-purpose CPU, but special-purpose hardware may also be used (and may be required for future higher-throughput gateways).

A gateway is connected to two or more networks, appearing to each of these networks as a connected host. Thus, it has a physical interface and an IP address on each of the connected networks. Forwarding an IP datagram generally requires the gateway to choose the address of the next-hop gateway or (for the final hop) the destination host. This choice, called "routing", depends upon a routing data-base within the gateway. This routing data-base should be maintained dynamically to reflect the current topology of the Internet system; a gateway normally accomplishes this by participating in distributed routing and reachability algorithms with other gateways. Gateways provide datagram transport only, and they seek to minimize the state information necessary to sustain this service in the interest of routing flexibility and robustness.

Routing devices may also operate at the network level; in this memo we will call such devices MAC routers (informally called "level-2 routers", and also called "bridges"). The name derives from the fact that MAC routers base their routing decision on the addresses in the MAC headers; e.g., in IEEE 802.3 networks, a MAC router bases its decision on the 48-bit addresses in the MAC header. Network segments which are connected by MAC routers share the same IP network number, i.e., they logically form a single IP network.

### Internet Gateway Requirements

An Internet gateway is an IP-level router that performs the following functions:

1. Conforms to specific Internet protocols specified in this document, including the Internet Protocol (IP), Internet Control Message Protocol (ICMP), and others as necessary.
2. Interfaces to two or more packet networks. For each connected network the gateway must implement the functions required by that network. These functions typically include:
   - encapsulating and decapsulating the IP datagrams with the connected network framing (e.g., an Ethernet header and checksum);
   - sending and receiving IP datagrams up to the maximum size supported by that network, this size is the network’s "Maximum Transmission Unit" or "MTU";
   - translating the IP destination address into an appropriate network-level address for the connected network (e.g., an Ethernet hardware address);
   - responding to the network flow control and error indication, if any.
3. Receives and forwards Internet datagrams. Important issues are buffer management, congestion control, and fairness.
   - Recognizes various error conditions and generates ICMP error and information messages as required.
   - Drops datagrams whose time-to-live fields have reached zero.
   - Fragments datagrams when necessary to fit into the MTU of the next network.
4. Chooses a next-hop destination for each IP datagram, based on the information in its routing data-base.
5. Supports an interior gateway protocol (IGP) to carry out distributed routing and reachability algorithms with the other gateways in the same autonomous system.
6. Provides system support facilities, including loading, debugging, status reporting, exception reporting and control.

### Resource Allocation

In order to perform its basic datagram-forwarding functions, a gateway must allocate resources; its packet buffers and CPU time must be allocated to packets it receives from connected networks, while the bandwidth to each of the networks must also be allocated for sending packets. The choice of allocation strategies will be critical when a particular resource is scarce. The most obvious allocation strategy, first-come-first-served (FCFS), may not be appropriate under overload conditions, for reasons which we will now explore.

A first example is buffer allocation. It is important for a gateway to allocate buffers fairly among all of its connected networks, even if these networks have widely varying bandwidths. A high-speed interface must not be allowed to starve slower interfaces of buffers. For example, consider a gateway with a 10 Mbps Ethernet connection and two 56 Kbps serial lines. A buggy host on the Ethernet may spray that gateway interface with packets at high speed. Without careful algorithm design in the gateway, this could tie up all the gateway buffers in such a way that transit traffic between the serial lines would be completely stopped.

Allocation of output bandwidth may also require non-FCFS strategies. In an advanced gateway design, allocation of output bandwidth may depend upon Type-of-Service bits in the IP headers. A gateway may also want to give priority to datagrams for its own up/down and routing protocols.

Finally, Nagle has suggested that gateways implement "fair queueing", i.e., sharing output bandwidth equitably among the current traffic sources. In his scheme, for each network interface there would be a dynamically-built set of output queues, one per IP source address; these queues could be serviced in a round-robin fashion to share the bandwidth. If subsequent research shows fair queuing to be desirable, it will be added to a future version of this document as a universal requirement.

### Broadcast and Multicast

A host which is connected to a network (generally a LAN) with an intrinsic broadcast capability may want to use this capability to effect multidestination delivery of IP datagrams.

### Address Resolution Protocol (ARP)

When sending out an IP packet, how is the destination Ethernet address determined?

ARP (Address Resolution Protocol) is used to translate IP addresses to Ethernet addresses. The translation is done only for outgoing IP packets, because this is when the IP header and the Ethernet header are created.

**ARP Table for Address Translation**

The translation is performed with a table look-up. The table, called the ARP table, is stored in memory and contains a row for each computer. There is a column for IP address and a column for Ethernet address. When translating an IP address to an Ethernet address, the table is searched for a matching IP address. The following is a simplified ARP table:
In summary, when the translation is missing from the ARP table, one IP packet is queued. The translation data is quickly filled in with ARP request/response and the queued IP packet is transmitted.

Each computer has a separate ARP table for each of its Ethernet interfaces. If the target computer does not exist, there will be no ARP response and no entry in the ARP table. IP will discard outgoing IP packets sent to that address. The upper layer protocols can't tell the difference between a broken Ethernet and the absence of a computer with the target IP address.

Some implementations of IP and ARP don't queue the IP packet while waiting for the ARP response. Instead the IP packet is discarded and the recovery from the IP packet loss is left to the upper layer protocols. This recovery is performed by time-out and retransmission. The retransmitted message is successfully sent out onto the network because the first copy of the message has already caused the ARP table to be filled.

The implementation of protocol P on a sending host S decides, through the operation of the TCP, that it must send a message to a host T which has an IP address

\[
\text{IP address: } 223.1.2.2
\]

The appropriate sender then sends an ARP request packet which includes the IP address

\[
\text{IP address: } 223.1.2.2
\]

out on the network to every computer. The computer T responds with an ARP reply packet which includes the IP address

\[
\text{IP address: } 223.1.2.2
\]

and the Ethernet address of the sender of the ARP request. The Ethernet address is in the form

\[
\text{Ethernet address: } 08-00-10-99-AC-54
\]

The ARP module examines the Ethernet address field and sends the packet to the Ethernet driver. The Ethernet driver looks at the Type field and passes the ARP packet to the ARP module. The ARP module examines the Ethernet address field and adds the sender's IP and Ethernet addresses to its ARP table.

The updated table now looks like this:

<table>
<thead>
<tr>
<th>IP address</th>
<th>Ethernet address</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.2.1</td>
<td>08-00-39-00-2F-C3</td>
</tr>
<tr>
<td>223.1.2.2</td>
<td>08-00-28-08-39-A9</td>
</tr>
<tr>
<td>223.1.2.3</td>
<td>08-00-5A-21-A7-22</td>
</tr>
<tr>
<td>223.1.2.4</td>
<td>08-00-10-99-AC-54</td>
</tr>
</tbody>
</table>

In summary, when the translation is missing from the ARP table, one IP packet is queued. The translation data is quickly filled in with ARP request/response and the queued IP packet is transmitted.

Each computer has a separate ARP table for each of its Ethernet interfaces. If the target computer does not exist, there will be no ARP response and no entry in the ARP table. IP will discard outgoing IP packets sent to that address. The upper layer protocols can't tell the difference between a broken Ethernet and the absence of a computer with the target IP address.