CHAPTER 0

INTRODUCTION TO TCP/IP

This chapter gives an overview of TCP/IP networking principles that form the basis of discussion for many of the laboratories that are covered in this text. Using the example of a web access, the chapter gives some insight into the intricacies and complexities of TCP/IP networking. The chapter also provides an in-depth discussion of IP addresses and other addressing schemes used in the Internet.

TABLE OF CONTENTS

1. A TCP/IP NETWORKING EXAMPLE 1

2. THE TCP/IP PROTOCOL SUITE 8

2.1. AN OVERVIEW OF THE TCP/IP PROTOCOL SUITE 9

2.2. ENCAPSULATION AND DEMULTIPLEXING 12

2.3. DIFFERENT VIEWS OF A NETWORK 17

3. THE INTERNET 20

3.1. A BRIEF HISTORY 20

3.2. INFRASTRUCTURE OF THE INTERNET 21

3.3. ADMINISTRATION AND STANDARD BODIES OF THE INTERNET 24

4. ADDRESSES AND NUMBERS 26

4.1. MEDIA ACCESS CONTROL (MAC) ADDRESSES 26

4.2. PORT NUMBERS 28

4.3. IP ADDRESSES 28

4.3.1. SUBNETTING 30

4.3.2. CLASSFUL ADDRESSES 33

4.3.3. CLASSLESS INTER DOMAIN ROUTING (CIDR) 36

4.3.4. THE FUTURE OF IP ADDRESSES 38

5. APPLICATION LAYER PROTOCOLS 38

5.1. FILE TRANSFER 39

5.2. REMOTE LOGIN 42

5.3. ELECTRONIC MAIL 44

5.4. THE WEB 47

5.5. RECENT APPLICATIONS 52

1. A TCP/IP Networking Example

Consider a web browser (a web client) at a host with name *Argon.cerf.edu* (“*Argon*”) that makes a web access to a web server on a host with name *Neon.cerf.edu (“Neon”)*.[1] The web access is illustrated in Figure 0.1. Both hosts are connected to the Internet. The web access is a request for the home page of the web server with URL *http://Neon.cerf.edu/index.html.*

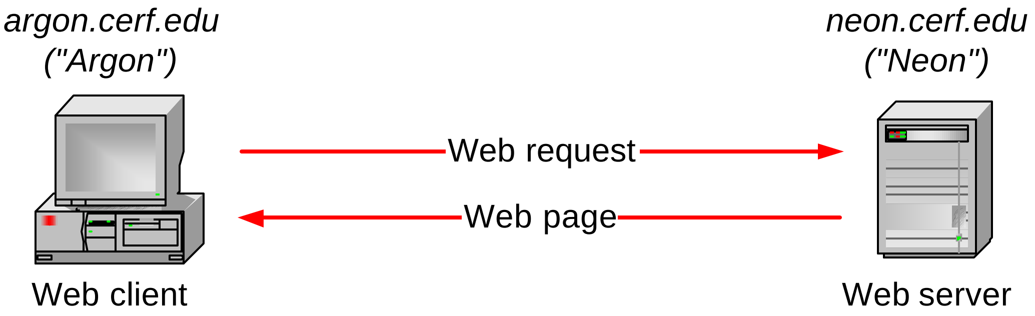


Figure 0.1. A simple web request.

We will explore what happens in the network when the web request is issued. We proceed, by following the steps that are performed by network protocols until the first packet from *Argon* reaches the web server at *Neon*. An outline of the steps involved is given in the table below. The example in this section is based on real traffic measurements, however, the names of the host have been changed.

A web client and a web server are programs that interact with each other using the Hypertext Transfer Protocol (HTTP). HTTP is a client-server protocol; the web client runs an HTTP client program and the web server runs an HTTP server program. When a web client requests a web page, the HTTP client sends an HTTP Request message to the HTTP server. When the web server finds the web page, the HTTP server sends the page in an HTTP Reply message.

HTTP uses the Transmission Control Protocol (TCP) to deliver data between the HTTP client and the HTTP server. When the HTTP client at *Argon* wants to send an HTTP request, it must first establish a TCP connection to the HTTP server at *Neon* (see Figure 0.2).

[1] Throughout we use the term *host* refers to a computer , or more generally, to any *end system* of communications.

1. Web client at Argon starts an HTTP Request.
2. Argon contacts its DNS server to translate the domain name “neon.cerf.edu” into IP address “128.143.71.21” and looks up the well-known port number of the web server (port 80).
3. The HTTP client at Argon requests a TCP connection to port 80 at IP address 128.143.71.21.
4. The TCP client at Argon requests its Internet Protocol (IP) to deliver an IP datagram with the connection request to destination 128.143.71.21.
5. The IP process at Argon decides that it cannot deliver the IP datagram directly, and decides to send the IP datagram to its default gateway 128.143.137.1.
6. The Address Resolution Protocol (ARP) at Argon sends an ARP request for the MAC address of IP address 128.143.137.1.
7. The ARP request is broadcast by the Ethernet device driver at Argon to all devices on the Ethernet network.
8. The router with IP address 128.143.137.1 responds with an ARP Response to Argon which includes MAC address 00:e0:f9:23:a8:20.
9. The IP process at Argon asks its Ethernet device driver to send the IP datagram in an Ethernet frame to MAC address 00:e0:f9:23:a8:20.
10. Ethernet device driver at router with MAC address 00:e0:f9:23:a8:20 unpacks the IP datagram, and passes it to its IP process.
11. The IP process at the router decides that it can deliver the IP datagram with destination 128.143.137.21 directly (without the need of additional routers).
12. The Address Resolution Protocol (ARP) at the router sends an ARP request for the MAC address of IP address 128.143.137.21.
13. The ARP request is broadcast by the Ethernet device driver at the router to all devices on the Ethernet network.
14. Neon (which has IP address 128.143.137.21) responds with an ARP Response to the router which includes MAC address 00:20:af:03:98:28.
15. The IP process at the router asks its Ethernet device driver to send the IP datagram in an Ethernet frame to MAC address 00:20:af:03:98:28.
16. The Ethernet device driver at Neon unpacks the IP datagram contained in the Ethernet frame, and passes it to its IP process.
17. The IP process unpacks the TCP connection request contained in the IP datagram and passes it to the TCP server at port 80.
18. The TCP server at port 80 processes the TCP connection request.

Table 0.1. Overview of the steps of the networking example.

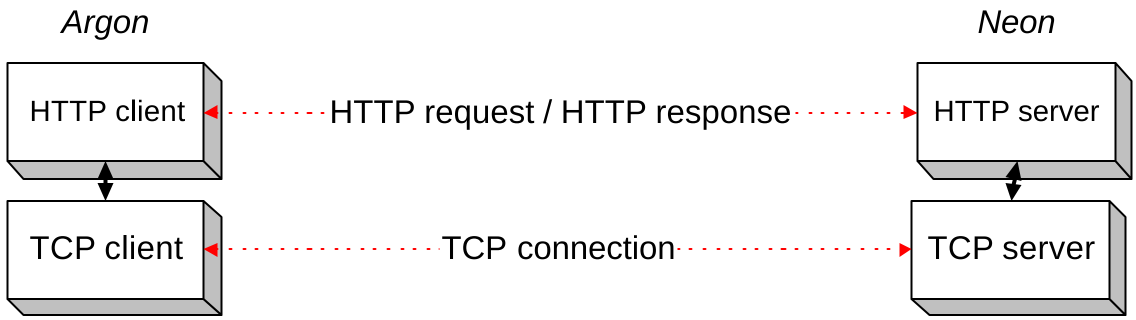


Figure 0.2 An HTTP session invoking a TCP/IP transmission

Before the HTTP client at *Argon* can ask the TCP protocol to establish a connection, it must resolve an addressing issue. The TCP protocol expects that addresses be written in terms of an IP address and a port number. An IP address is a 32-bit identifier that uniquely identifies a network interface connected to the Internet. Every network interface that is connected to the Internet has an IP address. The IP address of *Argon's* network interface is 128.143.137.144. Each byte (1 byte = 8 bits[2]) of the IP address is written as a decimal number, and the four numbers are separated by periods. A port number is a 16-bit address that can be used to identify an application program on a system. Together, an IP address and a port number can uniquely identify an application on the Internet.

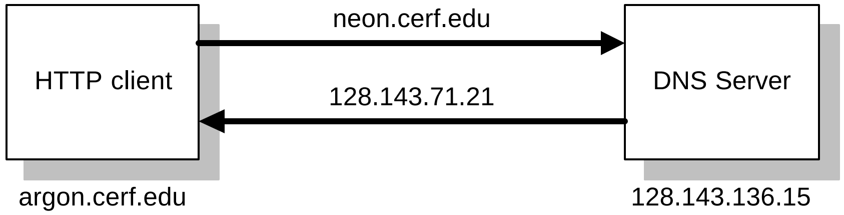


Figure 0.3. A DNS lookup for the IP address

Le us now see how the HTTP client at *Argon* determines the IP address and the port number of the web server at *Neon*. The IP address is determined using an Internet-wide address translation service, called the *Domain Name System (DNS),* that translates symbolic *domain names*, such as *Neon.cerf.edu,* into IP addresses, and vice versa. A host invokes the DNS service by sending a request to its DNS server. The location of the DNS server of a host is part of the network configuration of a host. Skipping over a lot of detail, *Argon* sends a query to its DNS server with the domain name "*Neon.cerf.edu*", and obtains in return the IP address for 128.143.71.21 (see Figure 0.3).

[2] Throughout this book, we use the convention that 1 byte is always 8 bits. The networking literature also uses the term "octet" to refer to a group of 8 bits.

Once the IP address is obtained, determining the port number of the HTTP server at *Neon* is simple. On the Internet, the server programs of widely used applications have permanently assigned port number, called *well-known port numbers*, and are stored in a configuration file of a host. The well-known port number of an HTTP server is port 80. When *Argon* contacts the HTTP server of *Neon* it assumes that the web server is reachable at port number 80.

Now, the HTTP client at *Argon* has the information needed to establish a TCP connection, and requests a TCP connection to port 80 at IP address 128.143.71.21. Like HTTP, TCP is a client-server protocol. The party that initiates the connection is called the TCP client. The party that is waiting for connection is called the TCP server. When a web server is started, the HTTP server sets up a TCP server that is waiting on the well-known port 80 for requests for TCP connections. The details of establishing a TCP connection a covered later in this book. Here we only note that the establishment of a TCP connection involves three steps: (1) the TCP client sends a TCP connection request to the TCP server, (2) the TCP server responds to the request, (3) the TCP client acknowledges this response and starts to send data. Here, we only follow the first step of the connection establishment procedure, where TCP sends a connection request to port 80 of IP address 128.143.71.21. We point out that this connection request does not contain any data. The HTTP request with the URL will be sent in the third step of the establishment procedure.

We now follow the steps that are involved to deliver the TCP connection request from *Argon* to *Neon.* The TCP client at *Argon* asks IP, the Internet Protocol, to deliver the connection request to IP address 128.143.71.21. IP takes the connection request, encapsulates it in an IP datagram (an IP datagram is the name of a packet in the Internet protocol), and delivers the IP datagram to *Neon.*

The network shown in Figure 0.4 gives an overview of what is involved in delivering the IP datagram from *Argon* to *Neon*. The figure shows that *Argon* and *Neon* are each connected to an Ethernet local area network. The picture shows that the *path (route)*, of an IP datagram from *Argon* to *Neon* goes through a router which connects the two Ethernet. A router, also called *IP* router or *gateway*, is a switching device with multiple network interface cards (NICs), which takes an IP datagram that arrives on one of its network interfaces and forwards the datagram on a different interface, with the intent to move the IP datagram closer to its destination. The forwarding decision is based on a routing table, which lists the name of a network interface for a set of destination addresses. Note in Figure 0.4 that the router has a domain name and an IP address for each of its network interfaces.

When *Argon* has the IP datagram for destination 128.143.71.21, it does not know the route that the IP datagram will take to its destination. The IP module at *Argon* merely determines if the IP datagram can be delivered directly to *Neon*, or if the datagram must be send to a router. A datagram can be delivered directly without the need for intermediate routers, if the sender and receiver are both connected to the same local network. *Argon* makes a decision whether *Neon* is on the same local network, by comparing the IP address of *Neon* with its own IP address. In this particular case, *Argon* assumes that IP addresses that match its own IP address

(128.143.137.144) in the first three bytes belong to interfaces that are on the same local network as *Argon*, and all other IP addresses are not on the same local network. With this criterion, *Neon* is on a different local network and *Argon* tries to forward the IP datagram to its *default gateway*. The default gateway of a host is the IP address of a router, and is part of the network configuration of any host. *Argon*'s default gateway is 128.143.137.1, with domain name *router137.cerf.edu ("Router137")*. Therefore, *Argon* sends the IP datagram to *Router137*.

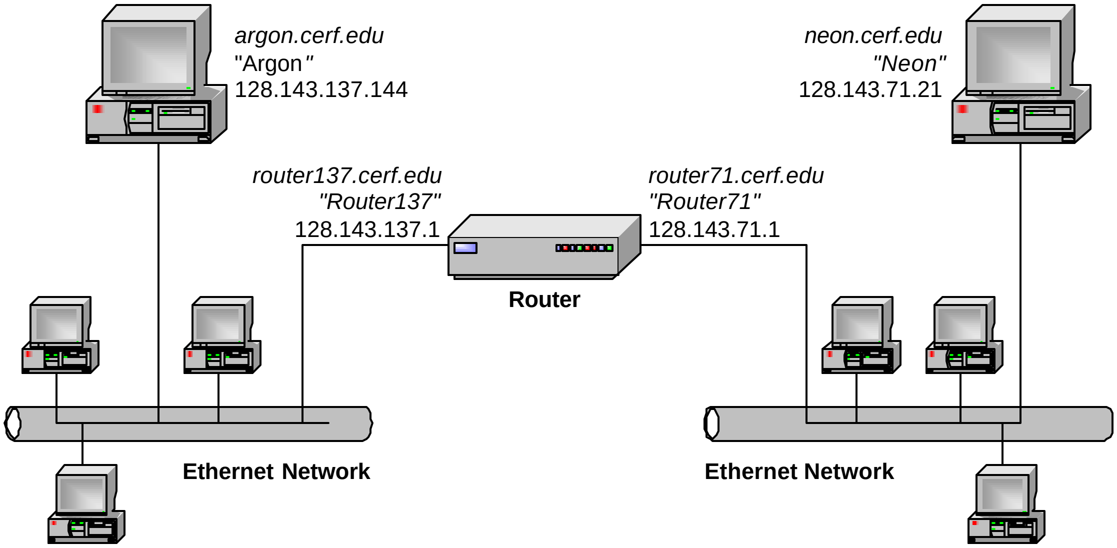


Figure 0.4. The route between *Argon* and *Neon*. The router in the center of the figure is referred to by any of the names associated its interfaces ("*Router137*" or "*Router71*").

To forward the IP datagram to the default gateway 128.143.137.1, *Argon* passes the IP datagram to the Ethernet device driver of its network interface card. The device driver will insert the IP datagram in an Ethernet frame (Ethernet packets are called *frames*), and transmit the frame on the Ethernet network. However, Ethernet frames do not use IP addresses, but have an addressing scheme that is based on 48 bit long MAC (media access control) addresses. MAC addresses are written in hexadecimal notation, where each byte is separated by a colon or a hyphen. For example, *Argon*'s MAC address is 00:a0:24:71:e4:44. Each Ethernet frame contains the MAC addresses of the source and the destination of the frame. Before IP can pass the datagram to the Ethernet device driver, it must find out the MAC address that corresponds to IP address 128.143.137.1. The translation between IP addresses and MAC addresses is done with the help of the Address Resolution Protocol (ARP).

The translation of addresses using ARP and the subsequent transmission of the frame that holds the IP datagram steps are shown in Figure 0.5. First, *Argon* sends out an ARP message of the form “What is the MAC address of 128.143.137.1?”. The ARP message is sent in an Ethernet frame that is broadcast to all Ethernet devices on the same network. When *Router137*, which has IP address 128.143.137.1, receives the ARP message, it responds with an ARP message that states “IP address 128.143.137.1 belongs to MAC address 00:e0:f9:23:a8:20”. Finally, when *Argon* receives this message, it passes the MAC address of the default gateway together with the IP datagram to the device driver and transmits the frame. In total, three Ethernet frames are transmitted on the Ethernet network. (We note that subsequent transmissions of IP datagrams from *Argon* to *Router137* do not require ARP messages. *Argon* keeps a list of known MAC addresses in a local table.)

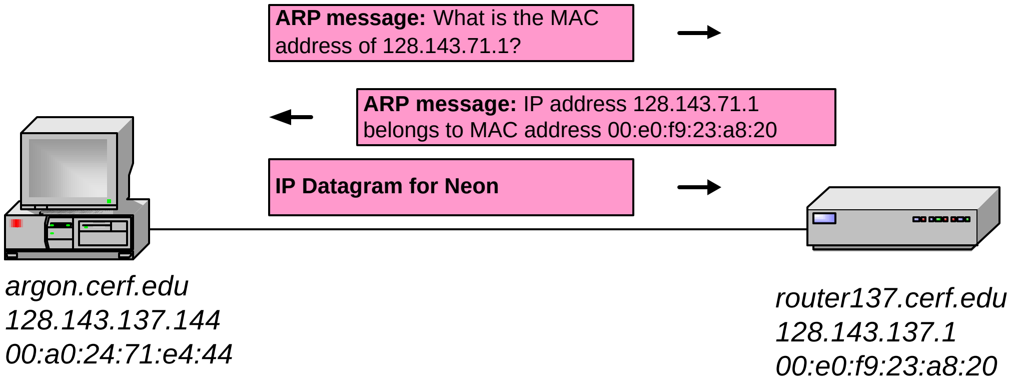

ARP message: What is the MAC address of 128.143.71.1?
ARP message: IP address 128.143.71.1 belongs to MAC address 00:e0:f9:23:a8:20
IP Datagram for Neon
argon.cerf.edu
128.143.137.144
00:a0:24:71:e4:44
router137.cerf.edu
128.143.137.1
00:e0:f9:23:a8:20

Figure 0.5. Ethernet frames transmitted to deliver the IP datagram to *router137*.cerf.edu.

When *Router137* receives the Ethernet frame from *Argon*, it first extracts the IP datagram and passes the datagram to the IP module. Once *Router137* receives the IP datagram from *Argon*, it makes a similar decision as *Argon*, that is, it determines if the datagram can be forwarded directly to *Neon* or it selects a router from its local routing table and forwards the packet to another router. *Router137* has multiple network interfaces, and, therefore, checks if the destination of the IP datagram is directly through <u>any</u> of its interfaces. To determine if a node is locally reachable, the router tries to match the first three bytes of the IP address of *Neon*, which is included in the IP datagram, to the IP addresses of its own interfaces: 128.143.137.1 and 128.143.71.1. Since a match occurs for 128.143.71.1, the router tries to forward the IP datagram directly to *Neon*.

The transmission of the IP datagram from the router to *Neon*, goes through the same steps as the delivery of the IP datagram from *Argon* to the router. First, the router acquires the Ethernet address of *Neon*, by sending an ARP request on the Ethernet interface which is associated with IP address 128.143.71.21. *Neon* sends its MAC address, 00:20:af:03:98:28, in an ARP response message. As soon as the router receives the MAC address of *Neon*, it sends an Ethernet frame that contains the IP datagram to *Neon*. Again, three Ethernet frames are as shown in Figure 0.6.

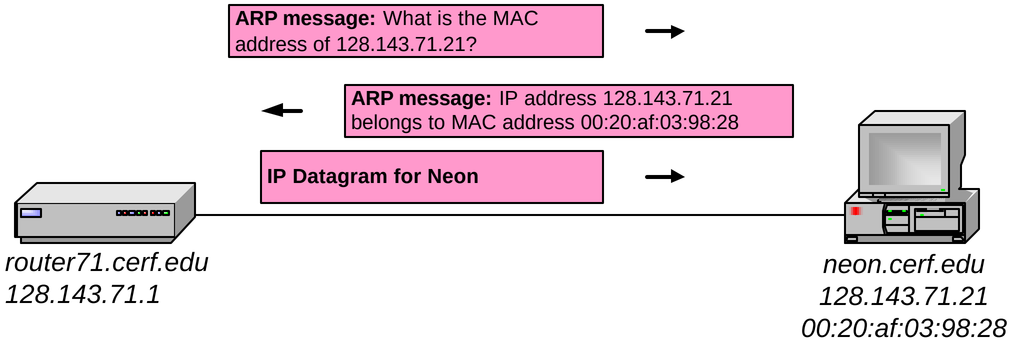

ARP message: What is the MAC address of 128.143.71.21?
ARP message: IP address 128.143.71.21 belongs to MAC address 00:20:af:03:98:28
IP Datagram for Neon
router71.cerf.edu
128.143.71.1
neon.cerf.edu
128.143.71.21
00:20:af:03:98:28

Figure 0.6. Ethernet frames transmitted to deliver the IP datagram from the router to *Neon*.

When *Neon* receives the Ethernet frame from *Router71*[3], it first extracts the IP datagram, and then delivers the TCP segment to the TCP server at port 80. At this time, the delivery of the first IP datagram from *Argon* to *Neon*, which contains the TCP connection request, is completed. What will happen next is that the TCP server at *Neon* responds to the TCP connection request, which involves the transmission of an IP datagram from *Neon* to *Argon*. The steps involved in delivering this IP datagram are identical to what we saw earlier. When the response is received at *Argon*, *Argon* can start to transmit the HTTP request.

The above description gives a good picture of the complexity of transferring data from one host to another. As you work your way through the chapters of this book, you will become familiar with each of the steps described above and you will understand the intricacies of this example.

[3] Note that from the perspective of *Neon*, the domain name of the router is *router71*.cerf.edu.

2. The TCP/IP protocol suite

The transport of data as shown in the example in the previous section is governed by a set of protocols that constitutes the TCP/IP protocol suite, also called *Internet protocol architecture* or Internet protocol stack. The protocols are organized in four layers, as shown in Figure 0.8: an application layer, a transport layer, a network layer, and a data link (or network interface) layer. Each protocol is assigned to one layer. At any given layer, a protocol requests the services of a protocol in a layer immediately below it, and provides services to a protocol in a layer immediately above it. The service requests generally relate to the delivery of data. This is shown in Figure 0.7, for the Web example from the previous section. When the HTTP client at *Argon* wants to send an HTTP request to the HTTP server at *Neon*, it requests from TCP the establishment of a TCP connection. Then TCP requests from IP the transmission of an IP datagram, which, in turn, requests from the Ethernet device driver the delivery of an Ethernet frame. Not shown in the figure is that the Ethernet device driver requests transmission from the MAC layer, which, in turn, requests the transmission of a bit stream. Whenever a protocol at a certain protocol layer wants to send data to a remote peer at the same layer, it actually passes the data to the layer below it, and asks that layer to perform the delivery of data. In this way, data can be thought of as traveling vertically through the protocol stack.

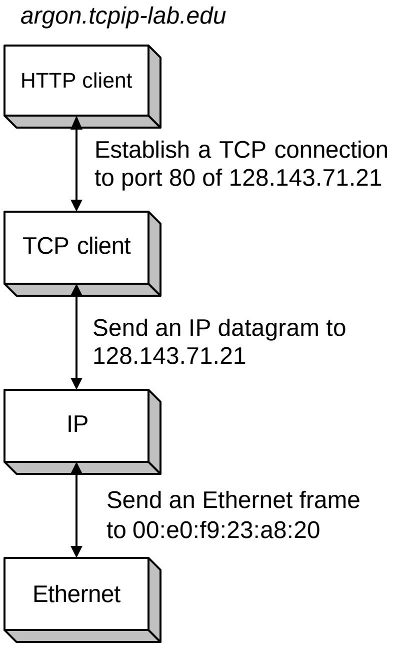


Figure 0.7. Invoking an IP connection to the remote host at IP address: 128.143.71.21

2.1. An Overview of the TCP/IP protocol suite

The TCP/IP protocol suite does not specify nor require specific networking hardware. The only assumption that the TCP/IP protocol suite makes is that the networking hardware, also called *physical network*, supports the transmission of an IP datagram. The lowest layer of the TCP/IP protocol suite, the data link layer, is concerned with interfacing to a physical network. Whenever a new network hardware technology emerges all that is needed to run Internet applications over the new ha rdware is an interface at the data link layer that supports the transmission of an IP datagram packet over the new hardware. This minimal dependence of the TCP/IP protocol suite on the underlying networking hardware contributes significantly to the ability of the Internet protocols to adapt to advances in networking hardware.

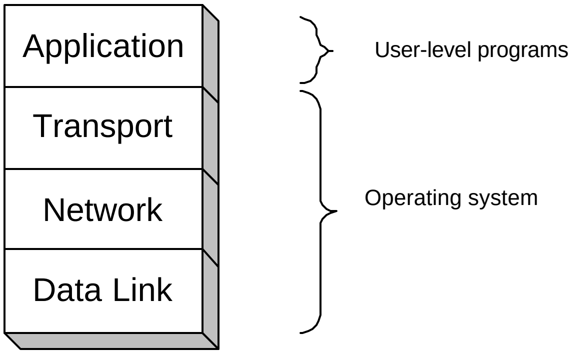


Figure 0.8. The Internet Protocol Stack

We next provide an overview of the layers of the TCP/IP protocol suite. At the *application layer* are protocols that support application programs. For example, the HTTP protocol is an application layer protocol that supports web browsers and web servers. Other protocols at the application layer are support of electronic mail (POP3, SMTP), file transfers (FTP), or command line interfaces to remote computers (Telnet, SSH).

The *transport layer,* also calle*d host-to-host layer,* is responsible for the exchange of application-layer messages between hosts. There are two transport-layer protocols in the TCP/IP protocol suite: the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP offers a reliable data service which ensures that all data from the sender is received at the destination. If data is lost or corrupted in the network, TCP will perform a retransmission. UDP is a much simpler protocol which does not make guarantees that delivery of data is successful.

The *network layer*, also called *internet layer*, addresses issues involved with forwarding data from a host to the destination across intermediate routers. All data transport at the network layer is handled by the Internet Protocol (IP). IP receives support from other protocols, such as the Internet Control and Messaging Protocol (ICMP) and the Internet Group Management Protocol

(IGMP) which perform error reporting and other functions. Also, IP relies on routing protocols, such as the Routing Information Protocol (RIP) or Open Shortest Path First (OSFP), or Border Gateway Protocol (BGP), which determine the content of the routing table at IP routers.

The *data link layer*, also called network access layer or interface layer, is concerned with sending and receiving data across either a point-to-point link or a local area network. There is a large variety data link layer protocols for local area networks and point to point links. Each data link layer protocol offers a hardware independent interface to the networking hardware. The functions of a data link layer protocol are partially implemented in hardware (on the network interface card) and partially in software (in the device driver). Most local area networks are broadcast networks where the transmission of a packet can be received by more than one host. In such networks, the data link layer has a sub-layer, which is called the Media Access Control (MAC) layer. The MAC layer can be thought of as residing at the bottom of the data link layer. The MAC layer runs a protocol that determines when a station on the local area network has permission to transmit. The MAC layer is realized in hardware on the network interface card and is not visible at the network layer or higher layers. Below the data link layer is the physical layer, which consists of the hardware that generates and interprets electrical or optical signals, and the cables and connectors that carry the signals.

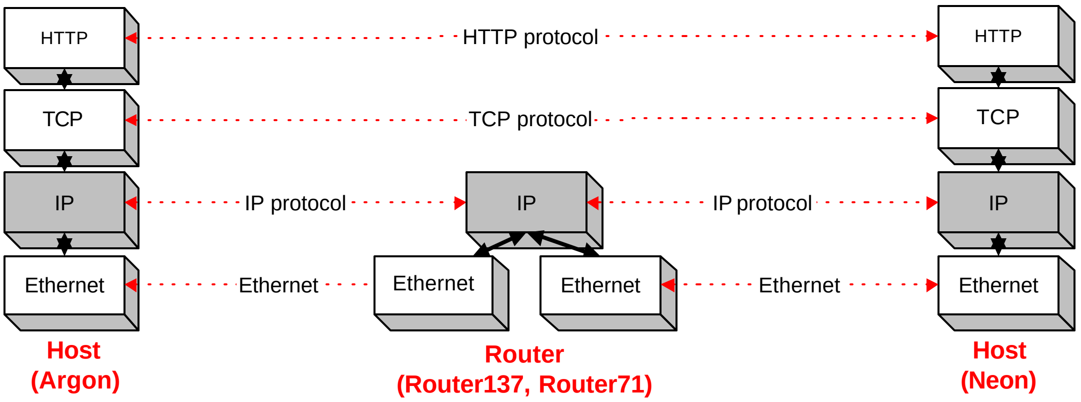


Figure 0.9. Protocols involved in data exchange between two hosts.

Figure 0.9 shows the protocols that are involved in the web access example between *Argon* and *Neon*. At each host, the figure shows protocol instantiations (*entities*) for HTTP, TCP, IP and Ethernet, where the Ethernet entity encompasses the data link layer, including the MAC layer, and the physical layer. At each layer, a protocol entity interacts with an entity of the same layer at a different system. The HTTP client at *Argon* interacts with an HTTP server at *Neon*, the TCP client at *Argon* interacts with the TCP server at *Neon*, and the IP and Ethernet entities at the hosts interact with corresponding entities at the router. Figure 0.9 illustrates that the routers do not have entities at the transport and application layer. Protocols at these layers are end-to-end protocols, and are not aware of the existence or the number of routers between two hosts. Conversely, routers do not interpret messages at the application or transport layer.

In Figure 0.10 we show the assignment of all protocols discussed in this book to the layers of the TCP/IP protocol suite. Figure 0.10 is not complete and shows only a subset of the protocols used in the TCP/IP protocol suite. An arrow in the figure indicates how protocols request services from each other. For example, HTTP requests the services of TCP, which, in turn, requests the services of IP, and so on. Figure 0.10 shows protocols that we have not mentioned so far. SNMP, the Simple Network Management Protocol, is used for remote monitoring and administration of equipment in an IP network. With exception of the ping program, all applications shown in the figure use the services of TCP or UDP, or, as in the case of DNS, both.

RIP, OSPF and PIM are routing protocols which determine the content of the routing tables at IP routers. Interestingly, RIP sends routing messages with UDP, a transport layer protocol. ICMP is a helper protocol to IP, which performs error reporting and other functions. IGMP is used for group communications.

The ARP protocol, which translates IP addresses to MAC addresses, so that IP can send frames over a local area network communicates directly with the data link layer.

Figure 0.10 clearly illustrates the central role of IP in the TCP/IP protocol suite. IP carries all application data, and can neither be bypassed nor replaced.

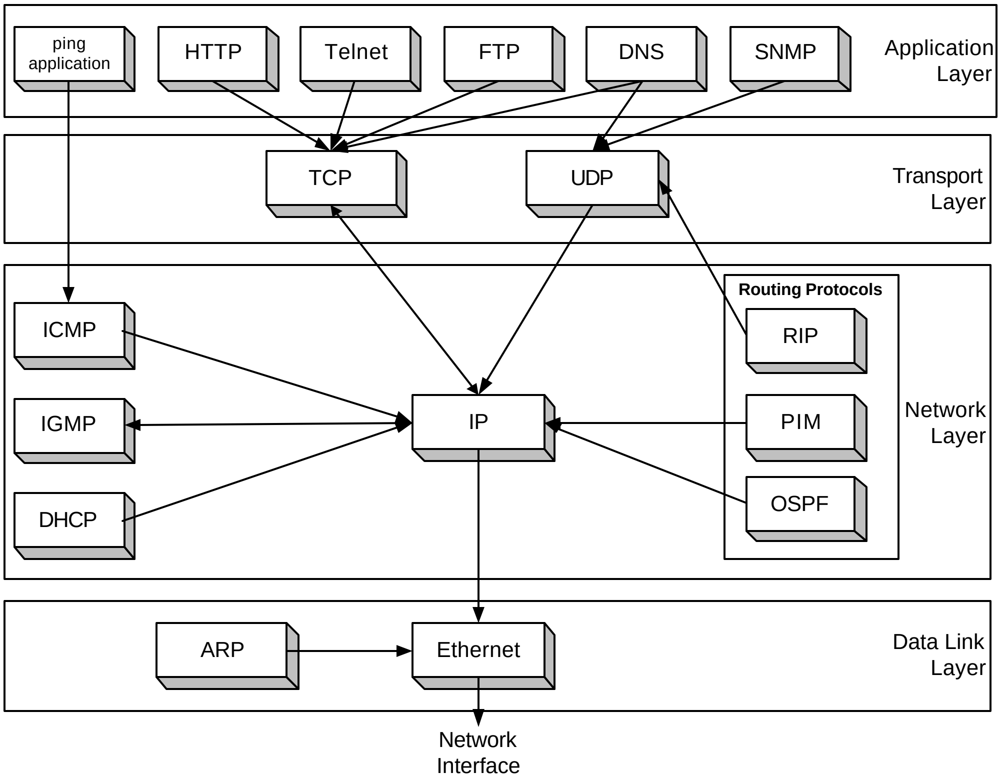


Figure 0.10. Assignment of Protocols to Layers.

2.2. Encapsulation and Demultiplexing

When data is passed from a higher layer to the next lower layer, a certain amount of control information is attached to the data. The control information is generally added in front of the data, and is called a header. If the control information is appended to the data, it is called a trailer. All protocols have headers and, but only very few have trailers. The process of adding header and trailers to data is called encapsulation. The data between the header and the trailer is called the *payload*. Together, header, payload, and (if present) trailer, make up a protocol data unit. There is a convention for naming protocol data units in the TCP/IP protocol suite: In TCP they are called *segments*, in UDP and IP, they are called *datagrams*, and at the data link layer, they are called *frames*. The generic name *packet* is used when the protocol data unit does not refer to a specific protocol or layer. The encapsulation process is illustrated in Figure 0.11. The protocol data unit at one layer makes up the payload at the next lower layer. Each layer adds a header and, in addition, Ethernet also adds a trailer. In this way, the overhead for sending application layer data increases with each layer.

Headers (and trailers) from a given layer are processed only by entities at the same layer. The header of a transmitted TCP segment is processed only by the TCP protocol at the host which receives the TCP segment. Each header of a protocol data unit is formatted into a number of protocol specific fields. The format of the header is part of the specification of a protocol. Header fields contain, among others, a source and a destination address, a checksum for error control, sequence numbers, and the length of the payload. If the header has not a fixed size, then the size of the header must be included as a field in the header.

At the receiver side, each layer strips off header fields from that layer and passes the payload to a protocol at the next higher layer. Each layer must decide to which higher-layer protocol to send the payload to. For example, an Ethernet device driver must assign the payload of a frame to IP or to ARP. Likewise, IP must assign the payload of an IP datagram to UDP or TCP or another protocol. This process of assigning the payload to a higher layer protocol is called *demultiplexing*. Demultiplexing is done using a field in the header of a protocol data unit, which identifies a higher layer protocol or an application process. Each protocol has such a demultiplexing field.

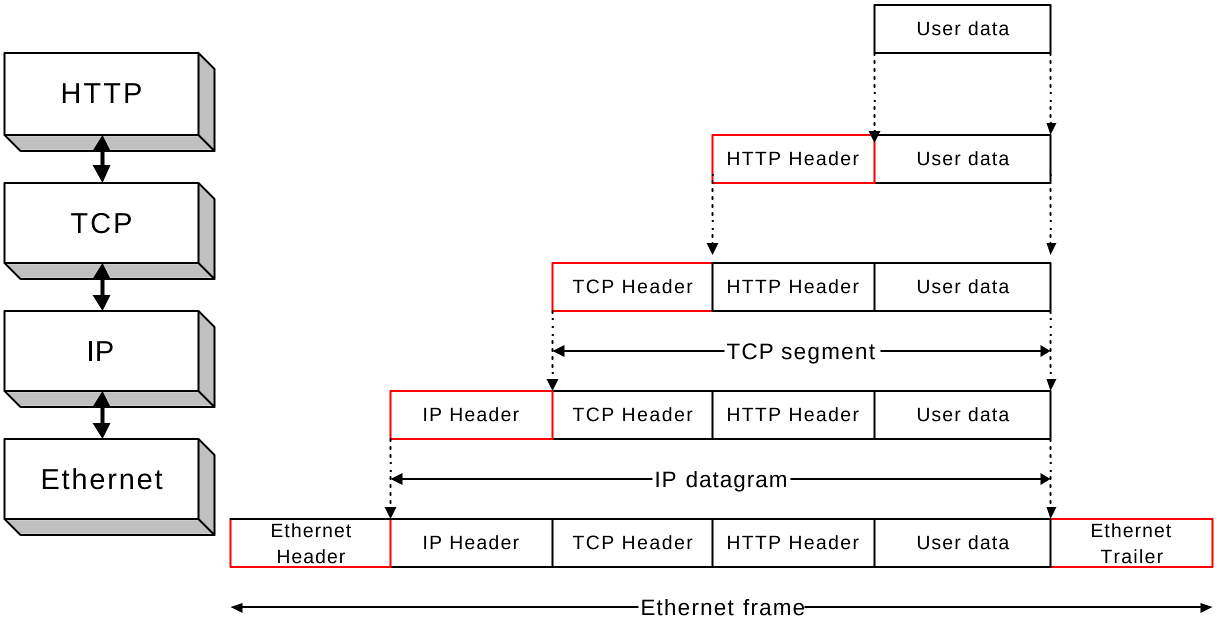


Figure 0.11. Encapsulation in the Internet.

Next we investigate encapsulation and demultiplexing by completely parsing an entire Ethernet frame. Specifically, we use the Ethernet frame transmitted by *Argon* to *Router137*, which contains the TCP connection requests for *Neon*. In hexadecimal notation the frame has a length of 58 bytes and looks like this:

```
00e0 f923 a820 00a0 2471 e444 0800 4500 002c 9d08 4000 8006 8bff
808f 8990 808f 4715 065b 0050 0009 465b 0000 0000 6002 2000
598e 0000 0204 05b4
```

Here, each byte is represented by two hexadecimal digits. The data shown does not contain the trailer of the Ethernet frame. In Figure 0.12 we show how these bytes are mapped to the Ethernet header, IP header, and TCP header.

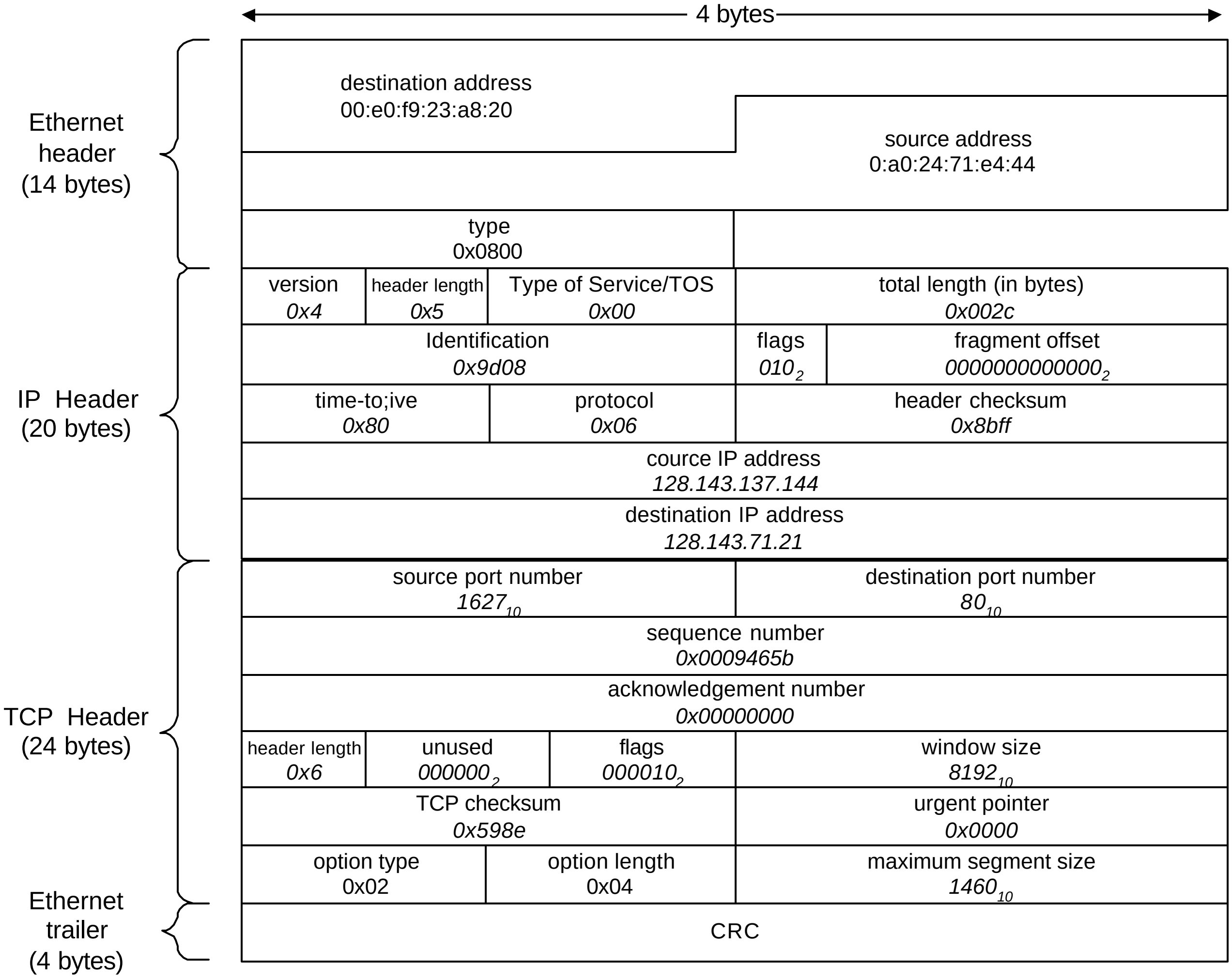


Figure 0.12. Encapsulation of a frame (0xN indicates a hexadecimal representation, N$_{10}$ indicates a decimal representation, and N$_2$ indicates a binary representation of a number).

We now parse the bytes of the frame, just like the protocol stack at a host would parse the frame. For orientation, we refer to Figure 0.12, which contains the final results of the processing of the frame. The first 14 bytes represent the fixed-sized Ethernet header:

```
00e0 f923 a820 00a0 2471 e444 0800
```

The first six bytes of the header contain the destination MAC address for this frame, and the next six bytes contain the source MAC address. We can verify that these are the addresses of *Router137* (`00:e0:f9:23:a8:20`) and *Argon* (`00:a0:24:71:e4:44`). The last two bytes of the Ethernet header contain the demultiplexing field. The value `0x0800`[4] indicates that the payload of the frame following the Ethernet header is an IP datagram. Therefore, the Ethernet device driver that receives the frame can pass the frame to IP for further processing.

The next 20 bytes are the IP header and are as follows:

```
4500 002c 9d08 4000 8006 8bff 808f 8990 808f 4715
```

The first four bits are the version number of the Internet Protocol. The value 0x4 indicates that this datagram is formatted according to the IP version 4 (IPv4) specification. The next four bits indicate the length of the IP header in multiples of four bytes. The value `0x5` states that the length of the IP header is 5 x 4 = 20 bytes, which is the minimum size of an IP datagram. The next byte specifies the type of service requested by this IP datagram. The value `0x00` states that this frame does not have any specific service requirements. The next two bytes (`0x002c`) contain the length of the IP datagram. At this point, IP knows that the IP datagram is 44 bytes long. The following two bytes (`0x9d08`) are an identifier which is assigned when the IP datagram is created. The next two bytes (`0x4000`) contain a 3-bit long flag field and a 13-bit long so-called fragment offset. These fields are used when an IP datagram is split into multiple pieces (*fragments*). If we use a binary representations for the two fields, then the flag field evaluates to 010_2,[5] which means that the second flag is set. This is the *Don't Fragment* flag, with the interpretation that the current IP datagram should never be split into multiple fragments. Accordingly, the value of the 13-bit long fragment offset is set to zero, or, in binary representation 0000000000000_2. The next byte is the Time-to-Live (TTL) field, which specifies the maximum number of routers that this IP datagram can cross. The value `0x80` states that the path of this IP datagram cannot exceed 128 routers. The next byte (`0x06`) is the protocol field, which is the demultiplexing field in the IP header. This field identifies the transport protocol that must process the payload following the IP header. The value `0x06` states that the payload of the IP datagram is a TCP segment. Following are a 2-byte long checksum field (`0x8bff`). The remaining bytes of the IP header contain the IP addresses of the source and destination. We can verify that `0x808f8990` corresponds to 128.143.137.144, the IP address of *Argon*, and that `0x808f4715` corresponds to 128.143.71.21, the IP address of *Neon*.

[4] When a number is preceded by a "0x" then the number is given in hexadecimal notation. Otherwise, the number is given in decimal notation.

[5] The subscript in 010_2 indicates that we use a binary representation.

The next 24 bytes are the header fields of the TCP segment:

```
065b 0050 0009 465b 0000 0000 6002 2000 598e 0000 0204 05b4
```

TCP parses the information in this header as follows. The first two bytes are the port numbers of the source (0x065b) and the destination (0x0050). Thus, the port number of the TCP client at *Argon* is 1627. This number was selected from a pool of available ports numbers when the TCP client was started at *Argon*. The destination port number is 80. As we already know, this is the well-known port number for HTTP servers. Note that the port numbers are the demultiplexing fields for a TCP segment, since they identify the protocol or application program at the next-higher layer. The next bytes are a 32-bit sequence number for data (0x0009465b) and a 32-bit sequence number for acknowledgements (0x00000000). The following four bits (0x6) provide the length of the TCP header in multiples of 4 bytes. Here, the TCP header is 6 x 4 = 24 bytes long. Since each TCP segment has a required size of 20 bytes, this value informs TCP that there is a 4 byte long option header following the required header fields. The six bits following the TCP header length are always set to zero: 000000_2. This seems to be a waste, but many protocols allocates certain bits in the header for "future use", and then never use these bits. The next six bits contain bit flags which contain important signaling information. The value for the fourteenth byte of the TCP header (0x02) we can derive the binary representation of the bit flags as 000010_2. We see that only flag number five is set. This flag is the SYN flag, which is set when a TCP client requests a TCP connection or when a TCP server responds to a request for a TCP connection . The following two bytes (0x200) indicate the size of the window for flow control purposes, and is set to 8192 bytes. The next two bytes of the header (0x598e) are a checksum. The two bytes following the checksum contain a 2-byte long urgent field. This field plays a role only when a certain flat, the URG flag, is set. In this segment, the flag is not set, and the urgent field is set to zero. The last four bytes (0x020405b4) contain an optional extension of the TCP header. Each TCP header option has three parts. The first byte (0x02) specifies the type of option, the second byte (0x04) the length of the option in number of bytes, and the remaining bytes contain the value of the option (0x05b4). The value of the type field identifies the option as a maximum segment size option, which sets the size of the largest TCP segment for the TCP connection. The value 0x05b4 sets the maximum size of a TCP segment to 1460 bytes.

We have now completely parsed the Ethernet frame. The result is summarized in Figure 0.12. It is interesting to note that there is no application data following the TCP header. The length of the IP datagram was given as 44 bytes, which is fully used by the IP header (20 bytes) and the TCP header (24 bytes). So, the entire frame consists only of header information.

Many exercises in the Internet Lab require the interpretation of protocol header information at different protocol layers. However, rather than going through the tedious exercise of parsing headers manually, the Internet Lab uses software tools that interpret the protocol headers and present them in an easily readable format.

2.3. Different Views of a Network

In all layers of the TCP/IP protocol suite the notion of a “network” is an abstract representation of the actual network environment. At the application and the transport layer, the network is seen as a single IP network that connects TCP clients and servers, as shown in Figure 0.13. Routers are not visible at this level of abstraction. The network abstraction does not distinguish whether applications are running on the same or on different hosts. As Figure 0.13 illustrates, the HTTP and TCP clients at *Argon* have the same view of the network when the HTTP server and TCP server are running locally.

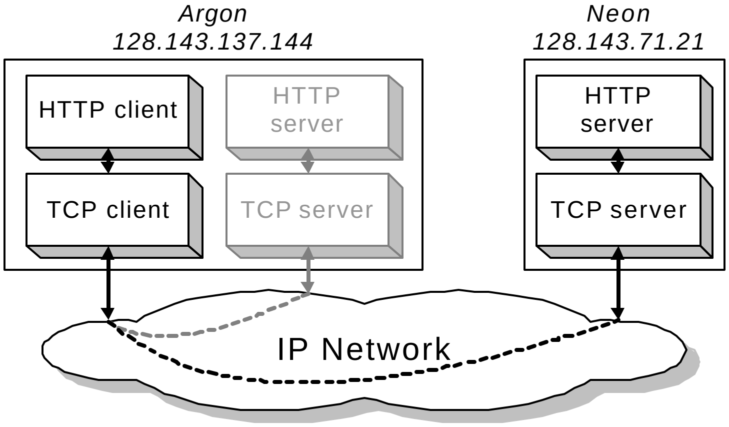


Figure 0.13 HTTP’s and TCP’s view of the “network”

Figure 0.14. IP’s view of a “network”

The network view at the IP layer is more detailed. Here the network is seen as an internetwork, that is, a collection of networks, often called *subnetworks* or *subnets*, which are connected by routers. In the web access example between *Argon* and *Neon*, the IP layer sees two networks which are connected by a single router, as shown in Figure 0.14.

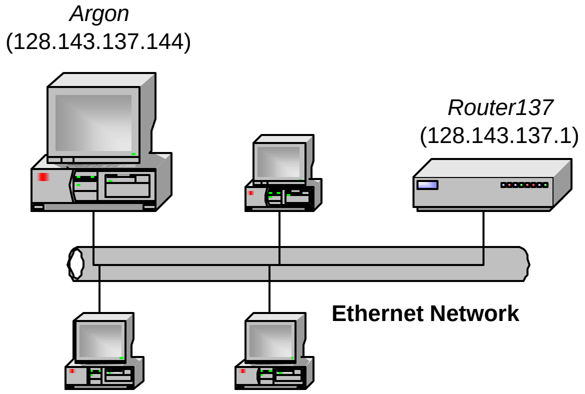


Figure 0.15. Ethernet's view of a "network".

At the data link layer, Ethernet views the network as a broadcast local area network, where multiple devices are attached to a shared bus, and each device on the bus can send frames to every other device on the bus. A single Ethernet network is also called an *Ethernet segment*. The example from Section 1, had two Ethernet networks, and we show one of the networks in Figure 0.15. The graphical representation of an Ethernet segment as a shared link tries to indicate the broadcast nature of an Ethernet segment, that is, each frame transmission can be received by all devices on the same segment, and multiple overlapping transmissions result in a collision.

After reviewing the network abstractions used by various layers, we now follow the actual path of the first IP datagram from *Argon* to *Neon* through the networking hardware. The networking hardware, through which the IP datagram is traveling, and their approximate location is shown in Figure 0.16. *Argon* and *Neon* are two PCs located on the same floor of the same building, the "IP Building", at Cerf University. The PCs are each connected to an Ethernet switch[6] in the basement of the IP Building. The Ethernet switches connect to a multiprotocol switch, which is partly an Ethernet switch and partly an ATM switch. The multiprotocol switch has a 155 Mbps fiber cable connection to an ATM switch. ATM (Asynchronous Transfer Mode) is a virtual-circuit packet switching method, that can be found in corporate or campus networks, as well as in wide-area networks that carry data, telephony, or video traffic.[7] For our purposes, it is sufficient to think of ATM as a data link layer protocol. ATM switches provide the backbone of the Cerf University campus network. The IP router in Figure 0.4, which we called *Router71* and *Router137*, is located in the TCP Building, which is across campus from the IP Building. The router is connected via an ATM link to the ATM switch in the TCP Building. The ATM switches in the TCP Building and the IP Building are connected by a 622 Mbps fiber link.

[6] Ethernet switches are the subject of Lab 5.

From Figure 0.16, it becomes very clear that none of the network abstractions of HTTP and TCP (Figure 0.13), IP (Figure 0.14), and Ethernet (Figure 0.15) allow an accurate description of the actual path of traffic between *Argon* and *Neon*. Even though the view of the IP layer suggests otherwise, the traffic between *Argon* and *Neon* does not take the shortest path. We see that all traffic between *Argon* and *Neon* leaves the building to reach the router in a different building, just to return to the building where the traffic originated. Scenarios as shown in Figure 0.16, i.e., where the path of traffic that is visible at the IP layer is very different from the actual flow of traffic, can be found very frequently.

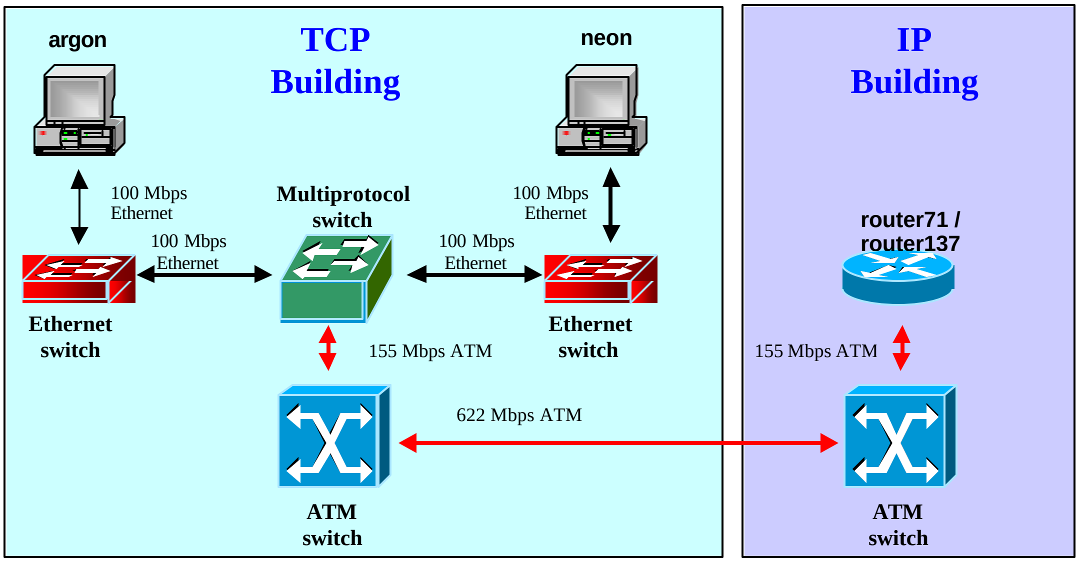
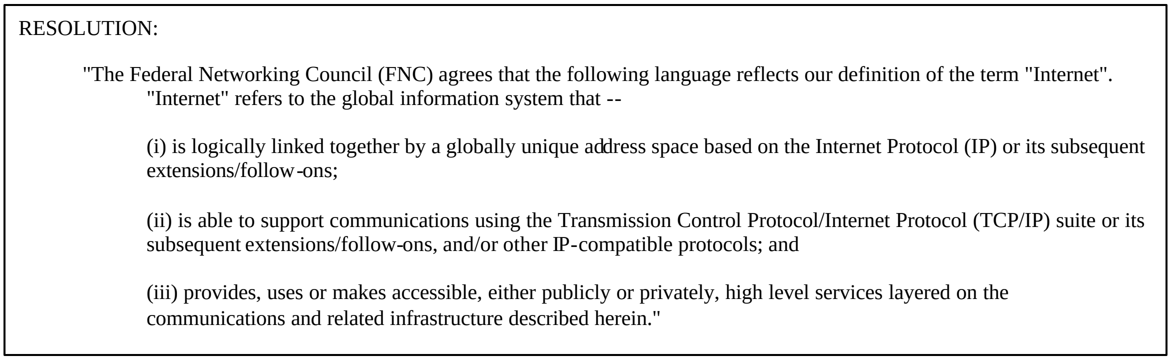


Figure 0.16. Network equipment.

3. The Internet

The term “Internet” refers to a global network of routers and hosts that run the TCP/IP protocol suite, and use the same global IP address space. Figure 0.17 shows an attempt for a precise definition of the Internet from 1995 by the United States Federal Networking Council.



RESOLUTION:

"The Federal Networking Council (FNC) agrees that the following language reflects our definition of the term "Internet".
"Internet" refers to the global information system that --

(i) is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons;

(ii) is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and

(iii) provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein."

Figure 0.17. Definition of the term “Internet” by the United States Federal Networking Council (FNC) (Source: http://www.ccic.gov/fnc/Internet_res.html).

3.1. A Brief History

The Internet started out in the late 1960s as a research network funded by the U.S. Department of Defense (DoD) in an effort to build a communication network based on the principles of packet switching. The network was initially referred to as the ARPANET (Advanced Research Projects Agency network). By the early 1970s, the emergence of several commercial and non-commercial packet-switching networks, and the development of various networking technologies, such as packet radio and local area networks, created a need to interconnect (“internetwork”) different types of networks through a common protocol architecture. The development of TCP, which initially included both the functionality of TCP and IP, in the mid-1970s met this need by providing a protocol that supports an *internetwork* or *internet* of networks, that use a variety of different technologies and are owned by different organizations. The term “the Internet” was used to refer to the set of all TCP/IP based internetworks with a common address space. By the early 1980s, the TCP/IP protocol architecture had taken the shape and form as described in Subsection 2.1, and had become the protocol standard for the ARPANET. Throughout the 1980s, new TCP/IP based network infrastructures emerged that connected universities and computing centers, and also interfaced to the existing ARPANET. One of these networks, the NSFNET, which was established by the National Science Foundation, became the successor of the ARPANET as the core infrastructure of the Internet. In the late 1980s, many regional and international TCP/IP based networks connected to the NSFNET, and created the foundation for today’s global Internet infrastructure. Today, the Internet consists of many thousand, mostly commercial, interconnected internetworks, and provides network services for more than 100 million hosts. The dramatic growth of the Internet is illustrated in Figure 0.18 which shows the number of hosts connected to the Internet since the early 1980s.

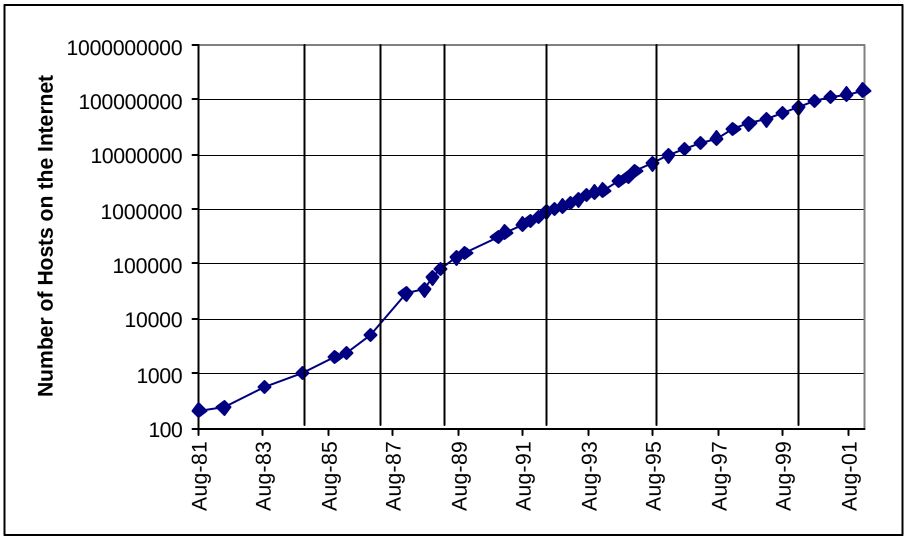


Figure 0.18. Growth of the Internet since the 1980s. The data is based on the Internet Domain Survey by the Internet Software Consortium

3.2. Infrastructure of the Internet

The infrastructure of the Internet consists of a federation of connected networks that are each independently managed. As is shown in Figure 0.19, the networks are organized in a loose hierarchy, where the attribute “loose” refers to the fact that the hierarchy is not required, but has evolved to its present form. At the lowest layer of the hierarchy are private networks, either campus or corporate networks, and local Internet service providers (ISPs). At the next level are regional networks which have a coverage of one or several states. At the top level of the hierarchy are backbone networks which span entire countries or even continents. ISPs, regional networks, and backbone networks are also called, respectively, Tier-3, Tier-2, and Tier-1 network service providers (NSPs). In the United States there are less than 20 Tier-1 NSPs, less than 100 Tier-2 NSPs, and several thousand Tier-3 ISPs.

Local ISPs, corporate networks, and ISPs are generally connected to one or more regional network or, less often, to a backbone network. The location where an ISP or corporate network obtains access to the Internet is called a Point-of-Presence (POP). The places where regional networks and backbone networks interconnect together for the purpose of exchanging traffic are called peering points. Peering can be done as public peering or private peering. Public peering is done at dedicated locations, called Internet exchange points (IXPs), where a possibly very large number of networks swap their traffic. In private peering, two networks establish a direct link to each other.

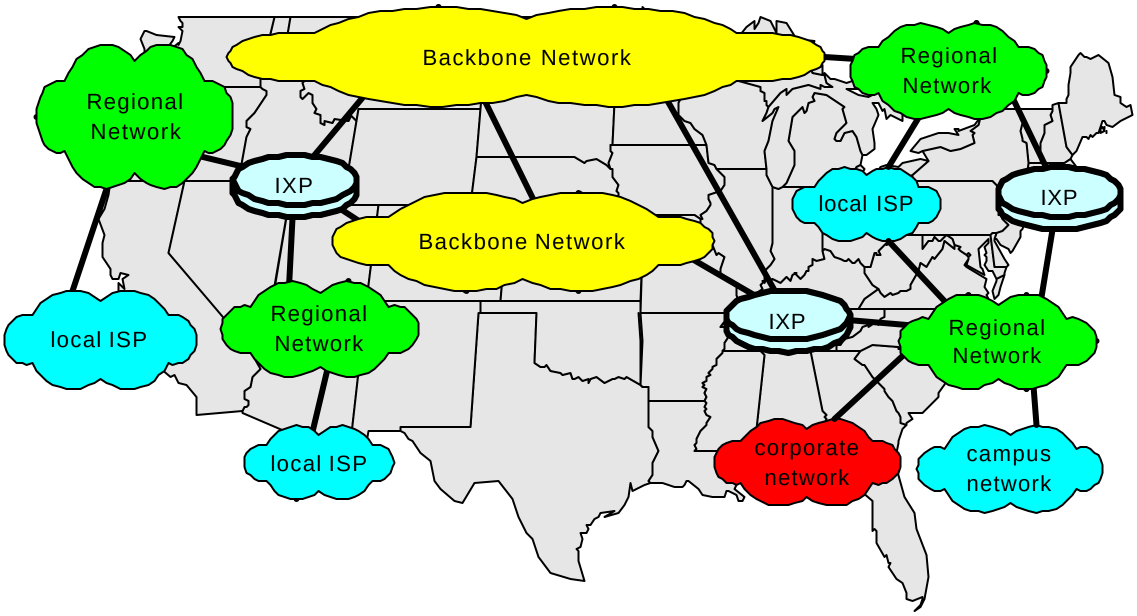
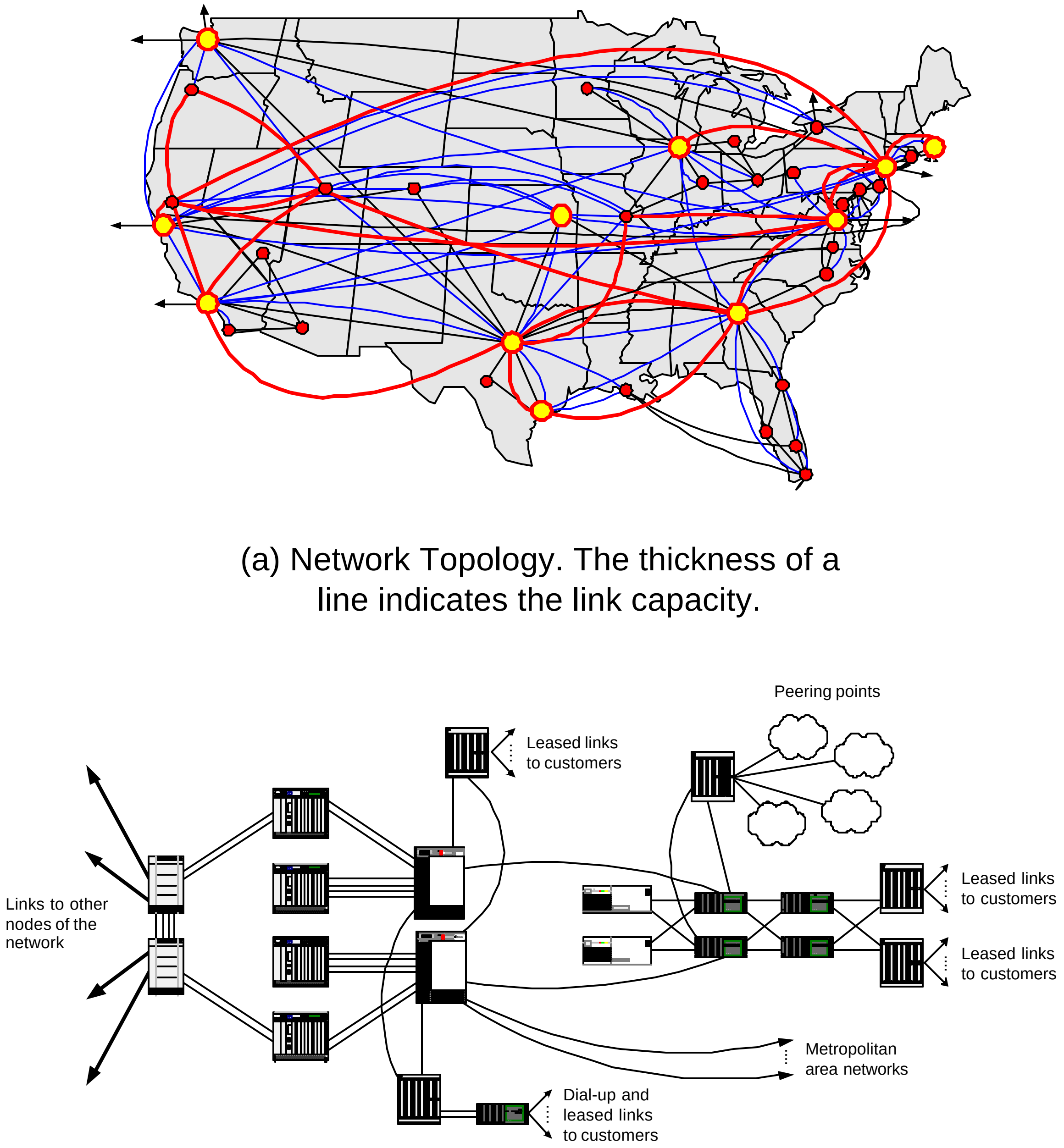


Figure 0.19. The organization of the Internet infrastructure.

The internal topology and equipment of each network of the Internet is independently maintained. The network topology of a local ISP may consist only of a small number of routers, a few modem banks for dial-in customers, and an access link to a regional network. The internal structure of a backbone network is significantly more complex, and is illustrated in Figure 0. 20. Figure 0. 20(a) depicts the network topology. Each node of the network topology is a network by itself, with a large set of networking equipment. This is indicated in Figure 0. 20(b). The depicted devices are router or other pieces of network equipment, and each arrow represents an outgoing network link. Only a small number of links connect to other nodes of the same backbone network, and most links connect to other networks or to customers.



(a) Network Topology. The thickness of a line indicates the link capacity.

(b) Networking equipment at a single node of a backbone network. Each arrow indicates an outgoing link.

Figure 0. 20. Backbone network.

3.3. Administration and Standard Bodies of the Internet

There are many administrative bodies that oversee and manage the growth of the Internet. As the Internet infrastructure itself, the administration of the Internet is based on cooperation and characterized by a minimum of central control.

The **Internet Society (ISOC)** is an international nonprofit professional organization that provides administrative support for the Internet. Founded in 1992, ISOC is the organizational home for the standardization bodies of the Internet. The **Internet Architecture Board (IAB) is a** technical advisory group of the Internet Society, and, among others, provides oversight of the architecture for the protocols and the standardization process. The **Internet Engineering Task Force (IETF)** is a forum that coordinates the development of new protocols and standards. The IETF is organized into working groups that are each devoted to a specific topic or protocol. Working groups document their work in reports, called Request For Comments (RFCs), which also built the basis for Internet standards. All specification of the Internet protocols or issues related to the TCP/IP protocol suite are published as RFCs. For example, IP is specified in RFC 791, TCP in RFC 793, ARP in RFC 826, and so forth. A small subset of published RFCs are put on a track to become Internet standards. Internet standards are going through a set of stages, from Internet draft to proposed standard, draft standard, and finally Internet standard. RFC's that are approved Internet standards are assigned an STD number. Currently, there are 61 designated Internet standards.

An important part of the administration of the Internet consists of the assignment and administration of unique identifiers. These include domain names (e.g., tpcip-lab.net), IP addresses (e.g., 128.143.71.21), well-known port numbers (e.g., port 80 for web servers), protocol numbers for demultiplexing (e.g., 0x60 to indicate that an IP datagram contains a TCP segment), and many others. From the early days of the ARPANET until the end of the 1980s, the administration of identifiers was done by a single person (Jon Postel from the Information Sciences Institute). The growth, increasingly international participation, and the privatization of the Internet has since led to a structure where identifiers are managed by organizations with broad international participation. Currently, a private non-profit organization, called **Internet Corporation for Assigned Names and Numbers (ICANN)** assumes the responsibility for the assignment of technical protocol parameters, allocation of the IP address space, management of the domain name system, and others. The management of IP address and domain name allocation is not done by ICANN itself, but by organizations that are authorized by ICANN. ICANN has granted a small number of Regional Internet Registries (RIRs) the authority to allocate IP addresses within certain geographical areas. Currently, there are three RIRs: APNIC (Asia Pacific Network Information Centre) for the Asia Pacific region, RIPE NCC (Réseaux IP Européens Network Coordination Centre) for Europe and surrounding regions, and ARIN (American Registry for Internet Numbers) for the Americas and Sub-Sahara Africa. The distribution of domain names is done by a large number of private organizations that are accredited by ICANN. An important role of ICANN in the domain name assignment is the establishment of new top-level domains (e.g., edu, us, jp).

4. Addresses and Numbers

In the networking example in Section 1, we encountered a set of different address schemes: domain names (e.g., *Argon*.cerf.edu), IP addresses (e.g., 128.143.137.144) and port numbers (e.g., 80), and MAC addresses (e.g., 00:a0:24:71:e4:44).

It is important to realize that domain names, IP addresses, and MAC addresses are not assigned to hosts or routers, but to their network interfaces. Since most Internet users work on hosts with only a single network interface, the distinction between a host and the network interface of a host is often irrelevant. This is different when working with routers or hosts with more than one network interface, so-called multi-homed hosts. In Section 1, the router in Figure 0.4 had two domain names, two IP addresses, and two physical interfaces. Depending on the point of view we referred to this router as "*router137*.cerf.edu" or as "*router71*.cerf.edu", with the interpretation, "the router that has a network interface with domain name `*router137*.cerf.edu'" or "the router that has a network interface with domain name `*router71*.cerf.edu'".

We now present some of the details of the address schemes for MAC addresses, IP addresses and port numbers. We discuss issues relating to domain names in Chapter 9.

4.1. Media Access Control (MAC) Addresses

MAC addresses are used to designate network interfaces at the data link layer. The MAC addresses are used in the frame header to identify the source and destination of a frame. Not every data link layer protocol requires the use of addresses. For example, when two routers are connected by a *point-to-point* link, such as a high-speed serial link, there is no need to use addresses in a frame to identify source or destination: every frame that is transmitted at one end of a point-to-point link goes to the device at the other end of the link. So, the sender of a frame on a point-to-point link knows where the frame is going to, and the receiver of a frame on a point-to-point link knows that the sender of the frame is the device at the other end of the link. Examples of data link layers for point-to-point links, which do not use addresses for network interfaces, are PPP (Point-to-Point Protocol) and SLIP (Serial Line IP).

In a local area network, multiple network interface access a broadcast transmission medium, which plays the role of a *shared link*. Each transmission on the shared link can be received by all network interfaces attached to the shared link. As a result, each network interface must have a unique address, and each transmitted frame must carry a destination address and a source address. The destination address is needed so that a network interface that receives a frame on the shared link can determine if it is the intended receiver of the frame. The source address in the frame header is needed to identify the sender of a frame.

Most local area networks adopt an address scheme that was developed by the IEEE 802 committee, a standardization body that has defined many standards for local area networks, including Ethernet, token ring, and wireless LANs. The address scheme is part of the MAC layer that we introduced in Subsection 2.1. The addresses in this scheme are 48 bits long and are known as MAC addresses, hardware addresses, or physical addresses. The convention for writing MAC addresses uses a hexadecimal notation, where each byte is separated by a colon or a dash. So, *Neon*'s MAC address (00:a0:24:71:e4:44 or 00-a0-24-71-e4-44[8]) corresponds to 00000000 10100000 0100100 01110001 1110001 01000100.

The MAC address of a network interface card is permanent and is assigned when the card is manufactured. Each network interface card receives a globally unique MAC address. When the network interface card of a host is replaced, the host is reached via a different MAC address. Since recently, some network interface cards provide the capability to modify the MAC address of a network interface card. However, changing the MAC address of a network interface card may result in address duplication. Therefore, when the MAC address is modified it is important to ensure that the assigned MAC address is unique on the local network.

The assignment of the MAC address space is administered by the IEEE. Each company that manufactures network interface cards obtains address blocks from the IEEE in the form of a 24-bit prefix. The manufacturer uses this prefix for the first 24 bits of the MAC addresses of its network interface cards. The last 24 bits are assigned by the manufacturer, who must ensure that the these bits are assigned without duplication. Once the address block is depleted, the manufacturer can request a new address block from the IEEE. Note, however, that one 24-bit prefix lasts for $2^{24} \approx$ 16.7 million interface cards. An interesting consequence of this assignment scheme is that the MAC address can be used to identify the manufacturer of an Ethernet card. For the MAC addresses in Section 1, we can determine that the network interface card of *Argon* and *Neon* were manufactured by the 3COM Corporation, since the prefixes 00:a0:24 and 00:20:af are assigned to the 3COM Corporation, and that the Ethernet interface of the router with MAC address 00:e0:f9:23:a8:20 was manufactured by Cisco Systems.

Other than identifying the manufacturer, the address space of MAC addresses is flat, meaning that MAC addresses do not encode any hierarchy or other structure that can be exploited for the delivery of frames. There are a few special addresses. The address ff:ff:ff:ff:ff:ff, that is, the address with all 48 bits set to '1', is designated as broadcast address. A frame where the destination address is the broadcast address is sent to all network devices on the local area network. For example, the Ethernet frames that carry the ARP request packets shown in Figure 0.5 and Figure 0.6 have the destination address set to the broadcast address. Addresses where the first 24 bits are set to 01:00:5e are multicast MAC addresses. Multicast addresses are intended for group communication, when a single sender wants to send a frame to a group of several, but not all, hosts, and are discussed in detail in Chapter 7.

[8] Leading zeros in a byte a dropped, e.g., 0a is written as a.

4.2. Port numbers

At the transport layer, TCP and UDP use 16-bit port numbers to identify an application process, which is either an application-layer protocol or an application program. Each TCP segment or UDP datagram carries the port number of the source and the destination in the packet header. The IP address, the transport protocol number (0x60 for TCP, 0x11 for UDP), and the port number of a packet uniquely identify a process on the Internet. When a packet is received by the transport layer, the transport protocol number and the port number provide the demultiplexing information to assign the data to the correct process. Since the number of the transport protocol is used in the demultiplexing decision, a host can have ports for TCP and ports for UDP with the same number. As an example, a UDP port 80 and a TCP port 80 are separate ports, which that can be bound different application processes.

Most network services and application programs on the Internet have a client-server interaction. When allocating port numbers the server programs of the client-server applications are assigned to *well-known port number*, meaning that all hosts on the Internet have default port numbers for their server programs. For example, the well-known port number for an HTTP server is port 80, and the well-known port numbers of other applications are port 21 for File Transfer Protocol (FTP) servers, port 25 for the mail servers that run the Simple Mail Transfer Protocol (SMTP), and port 23 for Telnet servers. All well known port numbers are in the range from 0 to 1023. The assignment of well-known port numbers to the server programs of application-layer protocols is generally stored in a file on a host. The assignment can be found in RFC 1700 (*STD 2)*. The advantage of using well-known port numbers is that client programs of application layer protocols which need to access a server program, can obtain the port number of the server simply via a local lookup.

The assignment of port numbers 1024 and higher is not regulated. Port numbers in this range are used by client programs of network services. Generally, client programs allocate a port number only for a short period of time, and release the port number as soon as the client program has completed its task. Because of the temporary allocation, port numbers 1024 and above are called *ephemeral port numbers*.

When developing software for application programs and application layer protocols for the Internet, programmers generally use the (Berkeley or BSD) socket application programming interface. A socket provides to the application programmer an abstraction that is similar to that of a file. An application program can open a socket, and then read from and write to a socket, and, once a task is completed, close a socket. By associating (binding) a socket to a port number, an application program gains access to the Internet address space.

4.3. IP Addresses

An IP address is an identifier for a network interface at the network layer. When the network interface is connected to the Internet, the IP address must be globally unique within the address space of the Internet. This means that no two network interfaces on the Internet can have the same IP address. In Lab 8 we will see that private networks (intranets) that interface with the Internet, but have hosts that are not directly connected to the Internet may need not have unique addresses. In Lab 9, we will learn about a method for dynamic assignment of IP addresses with a protocol called DHCP.

Throughout this book, we will we use the IP addresses from the Internet Protocol Version 4 (IPv4), which is the current version of the Internet Protocol. IP addresses in IPv4 are 32 bits long and are written in a so-called dotted decimal notation. In this notation, the value of each byte is represented as a decimal number, and the four numbers are separated by dots. So, address 128.143.137.144 corresponds to 10000000 10001111 10001001 10010000. The conversion from the binary to the dotted decimal notation is illustrated in Figure 0.21.

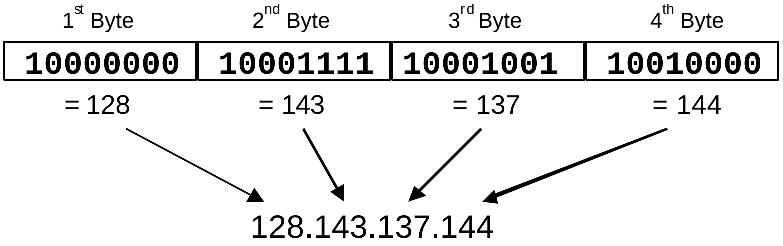


Figure 0.21. Derivation of the dotted decimal notation of an IP address

Each IP address consists of a network number which is followed by a host number.[9] The network number is also called the network prefix or, simply, the prefix. An organization obtains a network prefix either from its network service provider or directly from a Regional Internet Registrations (RIR). Once assigned a network prefix, the organization is free to assign host numbers to the network interfaces of routers and hosts on its network. The length of the network prefix is specified by a bitmask, called the *netmask*, which has a '1' in all bit positions of the prefix. In the so-called CIDR notation, the length of the network prefix is appended to the IP address, where IP address and the prefix length are separated by a '/' (slash). For example, the network of the Cerf University is assigned the 16-bit long network prefix 128.143. All hosts and routers on the Cerf University network have IP addresses in the range 128.143.0.0 – 128.143.255.255. The resulting netmask is 255.255.0.0. The CIDR notation for an IP address is 128.143.137.144/16, where /16 indicates the length of the prefix.

The separation of the address into a network number and a host number results in a two-layer hierarchy for IP addresses. When a router processes an IP address outside the scope of its

[9] Using "interface number" instead of "host number" would be more appropriate, but is not commonly used terminology. Note that a multi-homed host has one host number for each of its network interfaces.

network number, the router only looks at the network number and does not look at all at the host number.

Certain IP addresses have a special interpretation and cannot be used for network interfaces. All IP addresses where the host number is set to zero do not specify an interface but specify the address of a network. For example, the address 128.143.0.0/16 is the address of the Cerf University network. Many systems reserve address 0.0.0.0 to designate the *default route* in a routing table. An IP address where the host number has ones in all bit positions is a broadcast address. An IP datagram where the destination address is a broadcast is delivered to all hosts on the network. For example, the IP address 128.143.255.255 is the broadcast address for all hosts on the Cerf University network. The IP address 255.255.255.255 is called limited broadcast address, which tries to reach all hosts on the same local network. IP datagrams with this destination address are not forwarded by routers. By convention, the default router or gateway of a subnet has the host number set to one. The IP addresses of the router in Section 1, 128.143.137.1 and 128.143.71.1, followed this convention. Addresses with prefix 127.0.0.0/8 are reserved for *loopback interfaces*. The address of the loopback interface is used when the sender and receiver of an IP datagram reside on the same system, e.g., a web client accesses a web server on the same host. On most hosts, IP address 127.0.0.1 is reserved for the loopback address and is associated with the name *localhost*. When a host sends a datagram to IP address 127.0.0.1, the destination of the datagram is the local host, and the datagram is not transmitted on the network.

Then there are some IP addresses that are reserved for testing or experimental purposes. These are the addresses in the ranges 10.0.0.0 – 10.255.255.255, 172.16.0.0 – 172.31.255.255, and 192.168.0.0 – 192.168.255.255. When a system in the public Internet sees an IP datagram with an experimental IP address, the datagram is dropped. Experimental address ranges are often used for hosts that are not directly connected to the public Internet, e.g., hosts in testbed networks or private networks. We use experimental IP addresses in the Internet Lab. Therefore, if, for some reason, an IP datagram from the Internet Lab should leak to the public Internet, the datagram is designated as a packet that should be dropped and does not interfere with regular Internet traffic.

4.3.1. Subnetting

Since the networks of some organizations grow large, network operators can decide to subdivide the network into smaller subnetworks and assign each subnetwork its own network address. This process is known as *subnetting*. Subnetting is done by allocating some of the leading bits of the host number to indicate a *subnet number*. With subnetting, the network prefix and the subnet number make up an extended network prefix. The extended prefix can be expressed in terms of a subnetmask or, using CIDR notation, by adding the length of the extended subnetmask after the IP address. For example, for *Argon*, the first byte of the host number (the third byte of the IP address) is used to denote the subnet number[10]. With this, 128.143.0.0/16 is the IP address of the network, 128.143.137.0/24 is the IP address of the subnet, 128.143.137.144/24 is the IP address of the host, and 255.255.255.0 is the subnetmask of the host (see Figure 0.22).

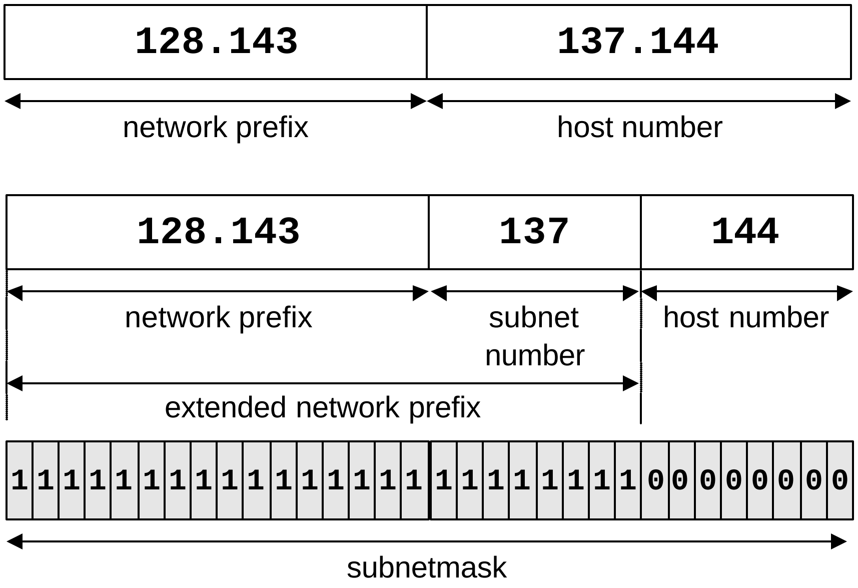


Figure 0.22. Network prefix and extended network prefix.

When a host or router holds an IP datagram it compares, for each of its network interfaces, its own IP address and the destination IP address. If the addresses match on the bits that correspond to the extended network prefix, the host or router will attempt to deliver the IP datagram directly. Otherwise, it will pass the IP datagram to another router.

The use of subnetting and the length of the subnet number in a network are up to the administrator of a network. The subnet structure is not visible outside the network where it is used. Hosts and routers do not know if and how subnetting is used on other networks. It is even possible to have different subnetmasks on the same network.

In Figure 0.23, Figure 0.24, and Figure 0.25 we illustrate different uses of subnetting Figure 0.23 shows a network without subnetting. Here the subnetmask of each host is identical to the network prefix. All hosts of the network believe to be part of the same subnetwork with IP address 128.143.0.0/16. In Figure 0.24, the hosts have an 8-bit long subnet number, resulting in

[10] In the real network, *Argon* does not use subnetting, and the (sub)netmask of *Argon* is 255.255.0.0. We will revisit our example with the correct subnetmask in a later chapter.

a 24-bit long extended network prefix or, equivalently, a subnetmask with value 255.255.255.0. Based on the IP addresses the hosts in the figure belong to subnet 128.143.137.0/24 or 128.143.71.0/24. Figure 0.25 shows an example where hosts on the same network have different subnetmasks. Starting from the left, the first two hosts have a 26-bit extended network prefix, the third host has a 24-bit extended prefix, and the fourth host has a 16-bit extended prefix. As a result, all hosts are on different subnets. Interestingly, the host with IP address 128.143.137.201 believes to be on the same subnet (with IP address 128.143.0.0/16) as all other hosts. However, based on their subnetmasks, any of the other hosts determines that 128.143.137.201 is not on its local subnet.

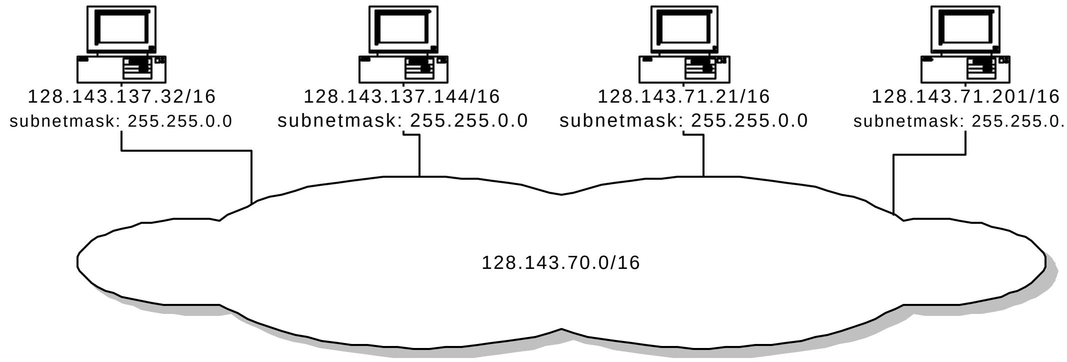


Figure 0.23. Network without subnetting.

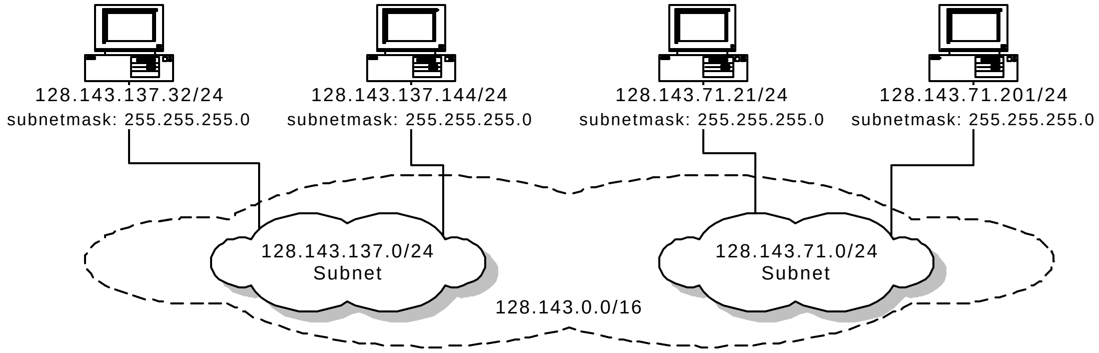


Figure 0.24. Network with subnetting.

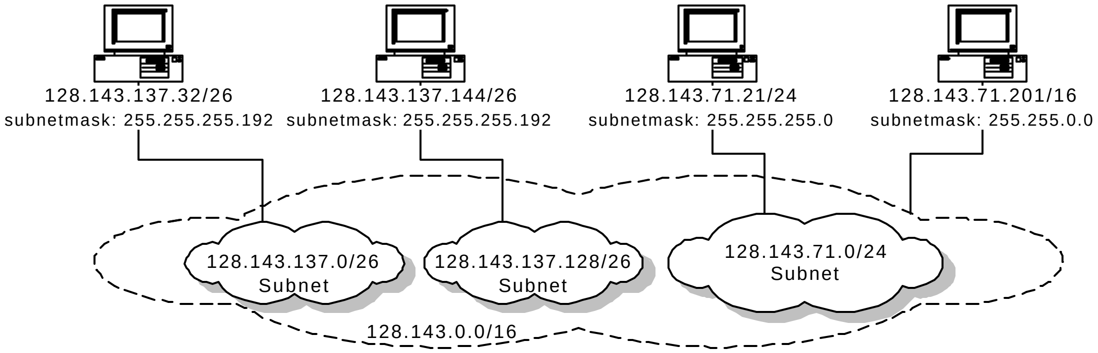


Figure 0.25. Network with different subnetmasks.

Subnetting leads to a three-layer hierarchy of IP addresses. The first level is the network number, the second level is the subnet number, and the third level is the host number. The delivery of an IP datagram follows the levels of this hierarchy. First, an IP datagram is delivered to the destination network. Once it arrives at the correct network, the datagram is delivered to the correct subnet, and then to the host on the subnetwork.

4.3.2. Classful addresses

In IPv4, the IP address space was partitioned into a five address classes: Class A, B, C, and D. IP addresses in Classes A, B and C are used for network interfaces.Class A addresses have the leftmost bit set to “0”. The network prefix of a Class A address is 8 bits long, resulting in a total of 2^7 different Class A networks. There are 24 bits available for the host number, resulting in up to 2^{24}-2 » 16.7 million hosts for each Class A network.[11] Each Class B address has the two leftmost bits set to “10”. Class B addresses have a network prefix that is 16 bits long. As result, the maximum number of Class B networks is 2^{14}=16,384. The maximum number of hosts in a Class B network is 2^{16}-2» 64,000. In Class C addresses, the three leftmost bits are set to “110”. Here, the network prefix has a length of 24 bits. The maximum number of Class C networks is approximately 2 million, and the number of hosts on a Class C network is limited by 2^8-2 = 254.

The motivation for the different classes is that a network administrator who wants to set up a new network requests a network prefix that matches the size of the network. Class A network addresses are intended for very large networks, class B addresses for networks that have a few thousand hosts, and Class C addresses are to be used for small networks. Note that classful addresses encode the class membership in the first bits of the IP address. Therefore, a router or host can derive the length of the network prefix by looking at the first bits of the IP address.

[11] This assumes that no subnetting is used and discounts the addresses where all bits of the host number are set to zero (network address) and all bits of the host number are set to one (broadcast address).

Class D and Class E addresses are not used to identify network interfaces. Class D addresses are reserved for multicast addresses, which are used to specify groups of hosts. A Class D address has the four leftmost bits set to “1110”. The details of Class D addresses will be discussed in Chapter 7. Class E addresses start with prefix “11110”. This address class was reserved for future use, but has never been defined. All address classes are illustrated in Figure 0.26. Using the dotted decimal notation the address ranges of the IP address classes are as follows:

Class A:	0.0.0.0 – 127.255.255.255
Class B:	128.0.0.0 – 191.255.255.255
Class C:	192.0.0.0 – 223.255.255.255
Class D:	224.0.0.0 – 239.255.255.255
Class E:	240.0.0.0 – 247.255.255.255

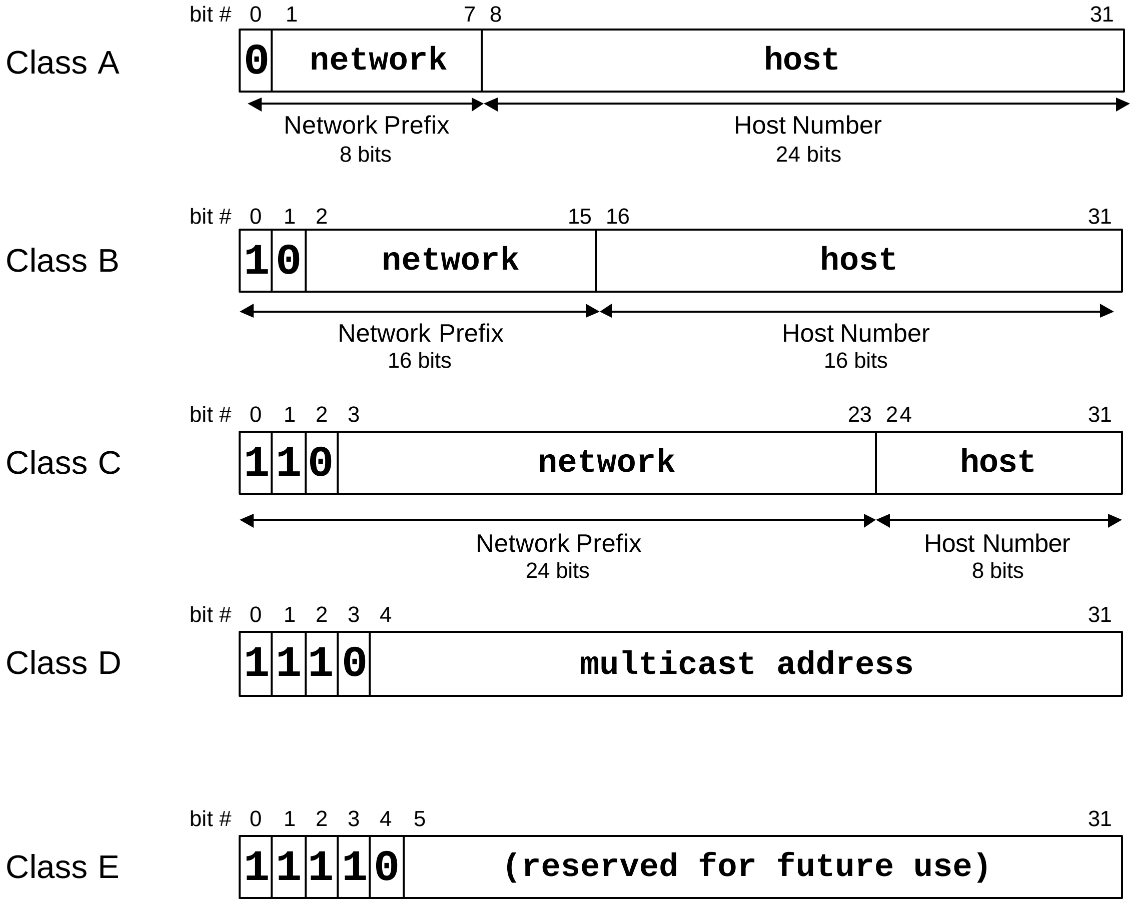


Figure 0.26. IP Address Classes. .

Classful addressing has a few inherent disadvantages. One disadvantage is a lack of flexibility when assigning network addresses. For example, when an organization wants to build a network for up to 10,000 hosts, the organization can either request a Class B address with more than 64,000 addresses or 40 Class C addresses which yields 10160 addresses. The former utilizes only a small fraction of the assigned portion of the address space, thus wasting a lot of IP addresses. The latter choice results in a large number of network addresses that need to be maintained by the network administrator. Ideally, the organization should allocate a network number with a network prefix with 14 bits, resulting in 2^{14}-2 = 16382 addresses.

The second problem is that the assignment of Class A, B, and C network addresses results in a flat address space and does not permit an aggregation of routing table entries. As a result, an IP router must keep one routing table entry for each network address. Since the number of networks, especially, Class C networks can be large, searching routing tables may become a performance bottleneck in a router.

In the early 1990s, the limitations of classful addressing became so apparent that the classful address scheme with fixed network prefixes was abandoned, and replaced with a scheme with flexible network prefixes. This scheme was called Classless Interdomain Routing (CIDR, pronounced “Cider”).

4.3.3. Classless Inter Domain Routing (CIDR)

When the fixed network prefixes and the flat address space of classful addressing seemed to limit the growth of the Internet, a new scheme for assigning IPv4 addresses was adopted which increased the efficiency of IP address assignments by using flexible network prefixes and reduced the routing table size via aggregation of routing table entries.

CIDR is a new interpretation of the IP address space. CIDR eliminates fixed-length prefixes, and, instead, allows network prefixes to have arbitrary lengths. With CIDR, the size of the network prefix must be provided with an IP address, since the length of the network prefix can no longer be inferred from the first bits of the IP address. Therefore, for each IP address, one must provide the length of the network prefix. This motivates the CIDR notation, where the network prefix length is added after a “/” that follows the IP address, e.g., as in 128.143.137.144/16. When subnetting is used, one writes the length of the extended network prefix. The CIDR notation allows to drop trailing zeros for network addresses, e.g., 128.143.0.0/16 can be written as 128.143/16.

With the variable size of the network prefix, IP address allocations for networks can be done more efficiently. Each prefix length corresponds to a block of addresses. For example, the prefix 10.0.112.0/20 describes an address block in the range 10.0.112.0–10.0.127.255. When an organization asks for a new network address it requests a prefix of appropriate length. For example, an organization that requires IP addresses for1000 hosts can request a /22 prefix. With a prefix of 22, 32 –22 = 10 bits are left for the host number, resulting in 2^{10} –2 = 1022 IP addresses.

The introduction of CIDR enables route aggregation in routing table entries. Essentially a routing table has two columns. The first column contains the destination IP address of a network and the second column contains the name of a network interface. With classful IP addresses, routing table entries are Class A, Class B or Class C addresses. A row with entry (128.143.0.0, interface #2) is interpreted as follows: When the router receives an IP datagram where the destination address has the (Class B) IP address 128.143.0.0/16, the router forwards this IP datagram to interface #2. With this scheme, a router must have one routing table entry for each network address. CIDR allows routing table entries to have a network prefix of any length. For example, a routing table entry of the form (128.0.0.0/8, interface #2) is interpreted, as follows. When the router receives an IP datagram where the destination address where the first 8 bits have value 128, the router forwards this IP datagram to interface #2. Therefore, a single routing table can contain routing information for many network addresses. This is called *route aggregation*. An issue with route aggregation is that a routing table can have multiple matches for the same destination. As an example, the following table has three matches for address 128.143.137.144:

Prefix	Interface
128.0.0.0/4	interface #5
128.128.0.0/9	interface #2
128.143.128.0/17	interface #1

With route aggregation in CIDR, one selects the routing table entry that matches the prefix of the destination address for the most number of bits. This is called *longest prefix matching*. In the above table, longest prefix matching selects the third row in the routing table, since the prefix 128.143.137.0/24 matches the prefix of 128.143.137.144 for 17 bits, as opposed to 4 and 9 bits, respectively, for the other entries.

Route aggregation is effective when the assignment of IP addresses is done in a hierarchical fashion. In a hierarchical address allocation, a backbone ISP obtains large block of IP addresses space from one of the RIRs, and then reallocate portions of this address blocks to its customers. If the customers are themselves network service providers, they divide their address blocks further and assign pieces of the blocks to their customers, and so on.

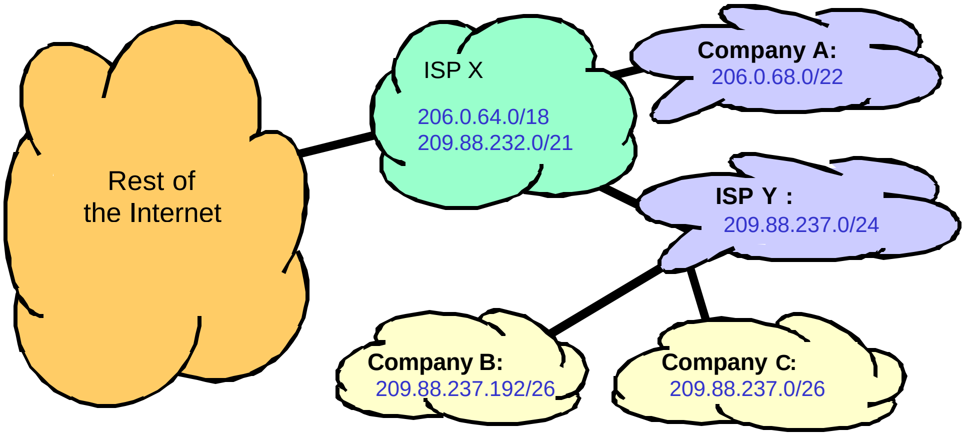


Figure 0.27. Address Space Allocation with CIDR.

In Figure 0.27, we show an example, where ISP X owns two IP address blocks with prefixes 206.0.64.0/18 and 209.88.232.0/21. The prefixes correspond to the IP address ranges 206.0.64.0 – 206.0.127.255 and 209.88.232.0 – 209.88.239.0. ISP X has allocated a /22 prefix (206.0.68.0/22) to Company A and a /24 prefix (209.88.237.0/24) to ISP Y. ISP Y, in turn, has given part of its allocated address space to Company B (209.88.237.192/26) and Company C (209.88.237.0/26).

In this example, the routers in the rest of the Internet (indicated by the cloud on the left of Figure 0.27), only require to have routing table entries for the prefixes 206.0.64.0/18, and 209.88.232.0/21 for traffic with a destination in ISP X or any of ISP X's customers. The routers need not know anything about the network addresses of ISP X's customers. When ISP X receives a packet which matches prefix 209.88.237.0/24, it forwards the packet to ISP Y. ISP X does not need to have routing table entries for the networks of Company B and Company C.

A disadvantage of route aggregation and the hierarchical organization of the IP address space is that network prefixes are tied to the organizations from which the prefixes are obtained. For example, when Company B decides that it wants to change its ISP, from ISP Y to ISP X, it cannot keep its IP addresses. Instead, Company B must return the allocated address block to ISP Y, request a new address block from ISP X, and modify the IP addresses on all systems in Company B.

4.3.4. The future of IP addresses

In principle, with 32-bit addresses, the Internet can have up to of 232 ≈ 4 billion IP addresses. However, since IP addresses are not assigned individually to hosts, but allocated in blocks to network service providers and organizations, which may, or may not, use all addresses of an allocated block, the Internet will eventually outgrow the available address space. Recognizing that the address space of IPv4 will eventually be depleted, the IETF has standardized a new version of the IP protocol, called IP Version 6 (IPv6). The length of an IP address in IPv6 is 128 bits, four times as large as an IPv4 address. With 128 bits it is not likely that IPv6 will ever out of IP addresses anytime soon. The address space is large enough to permit in the order of 10^{24} IP addresses per square inch on the surface of the Earth.[12]

5. Application Layer Protocols

The first generation of Internet applications from the late 1960s and early 1970s addressed the need to transfer files between computers, to have remote access to computing resources, and to support communication between users at different hosts. The applications, which met these needs: FTP, Telnet, and Email, were the predominant applications in the first decades of the Internet. The emergence of the World Wide Web in the early 1990s quickly made web browsing the most popular Internet application. Recently, applications for streaming audio and video, telephony over the Internet, and peer-to-peer file sharing have again changed the landscape of Internet applications.

This section discusses knowledge about applications interact with the Internet and how applications take advantage of the Internet to provide network services to end users.

[12] This is based on the Earth's surface, including land and ocean areas, having a size of approximately 196,935,000 square miles.

Many Internet applications are built on top of application-layer protocols that have been standardized by the IETF. In this section we briefly describe some of the most popular application layer protocols of the Internet, where we concentrate on the application protocols used in the Internet Lab. Since this book and the Internet Lab focus on the internal operations of the Internet protocols, and, less so, how applications use the Internet, we do not cover every detail of the applications and application-layer protocols. Rather, we provide an overview of the design of application-layer protocols, focusing on the interactions between application programs and application layer-protocols, and between application-layer protocols and transport-layer protocols. In pursuing this goal, we discover several similarities between different application layer protocols.

5.1. File Transfer

The **File Transfer Protocol (FTP)** for copying files between computer systems is one of the oldest application-layer protocols and was created before the TCP/IP protocol suite. FTP is a client-server protocol where an FTP client accesses an FTP server. The FTP client authenticates itself with a user name and a password to establish an FTP session. A successful authentication results in the establishment of an FTP session, where the FTP client can download ("get") and upload ("put") files and file lists. When transferring files, FTP respects the ownership and access privileges of files. Most hosts have a utility program, generally also called FTP, which provides an interactive command line interface to run an FTP session. This utility program is used throughout all parts of the Internet Lab. FTP clients can also be integrated in other applications, such as web browsers. Anonymous FTP is a special form of file transfer service that allows public access to files at an FTP server. An FTP client can establish an anonymous FTP session by supplying the user name "anonymous" and an arbitrary password (The FTP server usually requests to supply an email address as password).

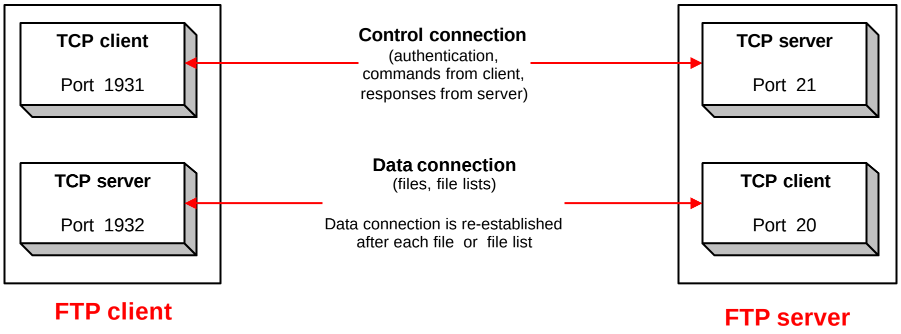


Figure 0.28. TCP connections in an FTP session.

FTP employs TCP as its transport protocol, thereby ensuring a reliable transfer of transmitted data. Two TCP connections are established for each FTP session, called control connection and data connection. The control connection is used for commands from the client and messages from the server. The data connection is used for the transport of files. As illustrated in Figure 0.28, an FTP server uses the well known TCP port 21 for the control connection, and the well-known TCP port 20 for the data connection, and an FTP client selects available ephemeral port numbers. The control connection is established at the beginning of the FTP session and stays up for the entire lifetime of the session. The control connection is used by the FTP client for authentication, for setting various session parameters, and for commands to download or upload files. The data connection is opened and closed for each transfer of a file or file. As soon as a file or a file list has been transferred, the data connection is closed. If there is another file transfer, the data connection is re-opened. By default, the data connection is established upon request by the FTP server. The FTP client starts a TCP server that waits for a connection on an ephemeral port, and sends the port number of this port on the control connection to the FTP server. After the FTP server receives the port, it can request a data connection to the FTP client.

A security concern with FTP is that the user name and the password submitted by the FTP client on the control connection at the beginning of an FTP session are not encrypted. Therefore, anyone with the ability to capture traffic from an FTP client can obtain the user name and password used by an FTP client.

In Figure 0.29 we show the interaction between an FTP client and an FTP server on the TCP control connection. The FTP client sends commands to the FTP server, and the FTP server responds with a three-digit response code and an explaining text message. The commands from the FTP client as well as the responses from the FTP server are transmitted as ASCII characters.[13] The end of a client command and the end of a server response is represented by an end-of-line sequence, which consists of the ASCII special characters Carriage Return (ASCII 10) followed by Line Feed (ASCII 13). When the TCP connection to TCP port 21 of the FTP server is established, the FTP server sends a message that it is ready to interact on a new FTP session. Then, the user supplies a user name and a password. If the authentication is successful, the user sends the IP address and port number of its ephemeral port for the data connection. The IP address and port number is sent in dotted-decimal notation, where the first four numbers specify the IP address and the last two numbers specify the port number. The string "192,168,123,175,7,140" in Figure 0.29 informs the FTP server that the client has a TCP server running on IP address 192.168.123.175 at TCP port 1932 (=7*256+140). The FTP server uses the IP address and port number for setting up the data connection. The command "RETR myfile" is a request by the client to copy the file "myfile" from the FTP server to the FTP client. By default, files are transmitted as text files using ASCII characters. However, the FTP client can change this default so that files are transmitted bit-by-bit without any interpretation. When the file transfer has been completed, the FTP server sends a message to the FTP client. At this

[13] ASCII (American Standard Code for Information Interchange) is an encoding format for textual data, which represents an alphanumeric character or special character by seven bits. Application-layer protocols transmit each ASCII character in a byte (octet) with the most significant bit set to zero. We provide a table which explains the ASCII code at the end of this chapter.

time, the FTP client can download or upload another files, or end the FTP sessions by issuing the command “Quit”.

```
Server:    220 ftp.local-lab.net FTP server (Version wu-2.6.1-16) ready.
Client:    USER student
Server:    331 Password required for root.
Client:    PASS MySeCreT
Server:    230 User root logged in.
Client:    PORT 192,168,123,175,7,140
Server:    200 PORT command successful.
Client:    RETR myfile
Server:    150 Opening ASCII mode data connection for myfile (61 bytes).
   // transmission of myfile on TCP data connection (66 bytes)
Server:    226 Transfer complete.
Client:    QUIT
Server:    221-You have transferred 66 bytes in 1 files.
Server:    221-Total traffic for this session was 400 bytes in 1 transfers.
```

Figure 0.29. Exchange of FTP control messages between FTP client and FTP server.

The **Trivial File Transfer Protocol (TFTP)** is a minimal protocol for transferring files without authentication and no separation of control information and data as in FTP. TFTP is frequently used by devices without permanent storage for copying an initial memory image (bootstrap) from a remote server when the devices are powered on. Due to the lack of any security features, the use of TFTP is generally restricted.

TFTP uses the unreliable transport protocol UDP for data transport. Each TFTP message is carried in a separate UDP datagram. The first two bytes of a TFTP message specify the type of message, which can be a request to download a file, request to upload a file, a data message, or an acknowledgement or error message. A TFTP session is initiated when a TFTP client sends a request to upload or download a file from an ephemeral UDP port to the (well-known) UDP port 69 of an TFTP server. When the request is received the TFTP server picks an ephemeral UDP port of its own and uses this port to communicate with the TFTP client. Thus, both client and server communicate using ephemeral ports.

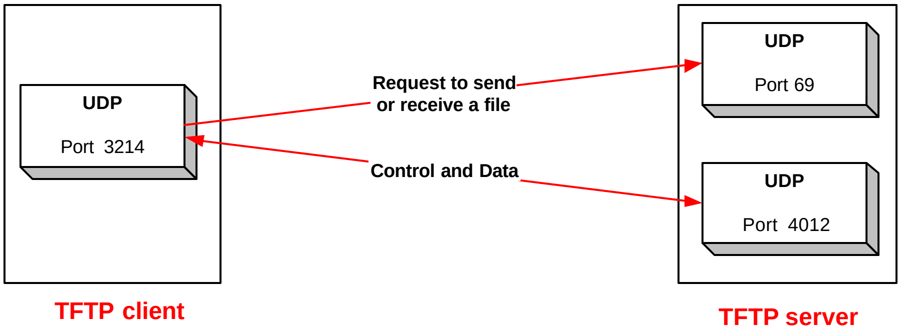


Figure 0.30. TFTP session.

Since UDP does not recover lost or corrupted data, TFTP is responsible for maintaining the integrity of the data exchange. TFTP transfers data in blocks of 512 bytes. Each block is assigned a 2 byte long sequence number and is transmitted in a separate UDP datagram. A block must be acknowledged before the next block can be sent. When an acknowledgment is not received before a timer expires, the block is retransmitted. When the receiver receives a block that is less than 512 bytes long, it assumes that the end of file has been reached.

5.2. Remote Login

Telnet is a remote login protocol for executing commands on a remote host. The Telnet protocol runs in a client-server mode and uses the TCP protocol for data transmission. A client initiates a Telnet session by contacting a Telnet server at a remote host. Similar to FTP, the origins of Telnet go back to the beginning of the ARPANET in the late 1960s. Recently, the use of Telnet in public networks has been discouraged since Telnet does not offer good protection against third parties that can observe ("snoop") traffic between a Telnet client and a Telnet server.

The mode of operation in a Telnet session is illustrated in Figure 0.31. At the Telnet client, a character that is typed on the keyboard is not displayed on the monitor, but, instead, is encoded as an ASCII character and transmitted to a remote Telnet server. At the server, the ASCII character is interpreted as if a user had typed the character on the keyboard of the remote machine. If the keystroke results in any output, this output is encoded as (ASCII) text and sent to the Telnet client, which displays it on its monitor. The output can be just the (echo of the) typed character or it can be the output of a command that was executed at the remote Telnet server.

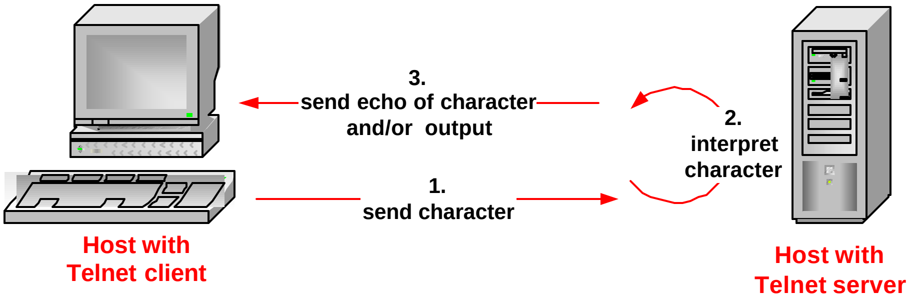


Figure 0.31. Client/server interaction in a Telnet session.

Telnet uses a single TCP connection for communications. The Telnet server uses the well-known TCP port 23 and the Telnet client uses an ephemeral TCP port. After establishing the TCP connection, the Telnet client and server negotiate a set of parameters for the Telnet session, including terminal type, line speed, if typed characters should be echoed to the client or not, and so on. Unless a Telnet session is explicitly configured not to do so, Telnet client sends one TCP segment for each typed character.

Telnet provides some independence from the differences of hardware and software at hosts by mapping input and output to a virtual device, called Network Virtual Terminal (NVT) or pseudoterminal. The Telnet client and Telnet server map the input from a keyboard and the output to a monitor to the ASCII character set. Thus, before a character is sent over the network it is encoded as ASCII character. Also, when an ASCII character is received on the TCP connection, the receiving host interprets the character and translates it into its local character format.

Rlogin is an alternative remote login application program for hosts that run the Unix operating system. Rlogin takes advantage of the fact that both the client and server run a similar operating system, and, for this reason, is simpler than Telnet. Rsh is an application program for the execution of a single command on a remote Unix host. There is also an application program for file transfers between Unix hosts, called rcp. However, this group of applications has poor security features, and is, therefore, often disabled.

The **Secure Shell** suite of protocols provides application layer services for remote login and file transfer services, similar to FTP, Telnet, rlogin, rsh, and rcp, but ensures secure encrypted communications between untrusted hosts. All components of Secure Shell provide authentication, confidentiality, and integrity of data, using a variety of encryption and authentication algorithms, and protect against common attacks on the security or integrity of communications between hosts. On the Internet, Secure Shell is run as a client-server protocol over a TCP connection and Secure Shell servers use the well-known TCP port 22.

5.3. Electronic Mail

Electronic Mail (email) is the primary Internet service for exchanging messages between users at different hosts on the Internet. The exchange of email messages is asynchronous, meaning that the transmission and retrieval of an email message can occur at different times.

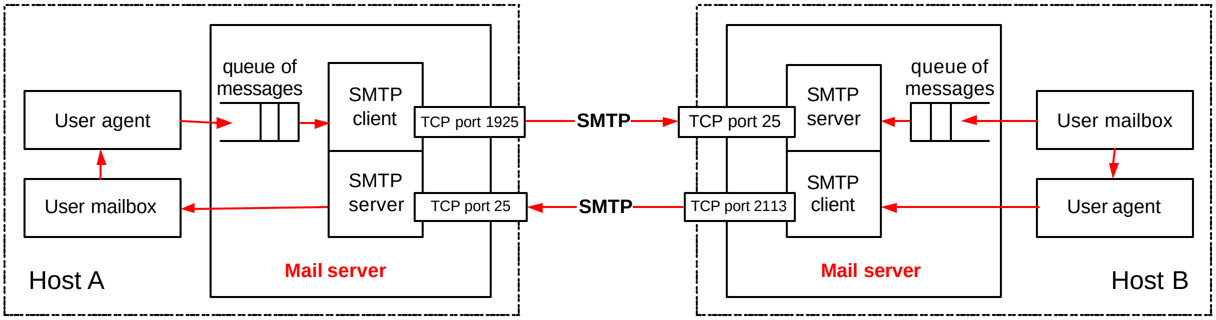


Figure 0.32. Components of an Email system.

The exchange of an email on the Internet is illustrated in Figure 0.32. A user on a host prepares an email on a mail preparation program, called a user agent. Examples of user agents on Unix operating systems are mail, xmail, elm, or pine[14]. Email messages are written in plain text and have the familiar structure shown in Figure 0.33, and users can add non-text files to an email message.

```
From: student@local-lab.net
To: instructor@remote-lab.net
Subject: Internet Lab
The dog ate my homework.
```

Figure 0.33. Email message.

The user agent passes the email to a mail server, where the message is queued for transmission. In Figure 0.32, the user agent and the mail server are on the same host, but it is also possible that they are on different hosts. The mail server uses the application-layer protocol SMTP (Simple Mail Transfer Protocol) to transmit the email message to the mail server of the receiver of the email. The sending mail server uses DNS, the domain name service, to locate the correct remote mail server. In addition to translating host names into IP addresses, DNS also provides the IP addresses of mail servers. For the email in Figure 0.33, the mail server at Host A queries DNS for the IP address of the mail server that is responsible for the domain of the email receiver, "remote-lab.net". Once the IP address of the remote mail server is obtained, the sending mail server starts an SMTP client and initiates a TCP connection to the SMTP server of the remote mail server at the well-known TCP port 25. As soon as the TCP connection is established, the SMTP client and server exchange SMTP commands and transfer the email message. All

[14] These are user agents for hosts with a Unix operating system.

commands and the email message are transmitted as plain text using ASCII characters. If an email message contains parts that are not text files, these parts are converted to ASCII characters before transmission.

The SMTP messages for transmitting an email from an SMTP client to an SMTP server is shown in Figure 0.34. As in FTP, all messages are transmitted as ASCII characters, using the end-of-line sequence to indicate the end of a command. The SMTP client issues commands to the server, and the server responds to each command with a three-digit response code and explaining text. After the TCP connection is established, the remote mail server sends a brief message. The client issues a HELO command followed by its domain name. Then, the client issues a MAIL FROM command sender's email address, which is followed by a RCPT TO command with the receiver's email address. After an acknowledgment from the server, the client issues a DATA command, and transmits the email message. By comparing the original message from Figure 0.33 to the actually transmitted message in Figure 0.34, we can see that the mail server has added several header lines to the message. When the entire email message is transmitted, the client indicates the end of the email by sending a line that only contains the character "." (period). When the SMTP server acknowledges the reception of the email message, the client can send another message or end the SMTP session. In Figure 0.34, the client issues a QUIT command to end the SMTP session. The mail server that receives an email message, adds the email to a file, called mailbox, which can be accessed by the receiver.

```
Server:   220 mailserver.remote-lab.net  Mail Server
Client:   HELO local-lab.net
Server:   250 Hello local-lab.net, pleased to meet you
Client:   MAIL FROM: student@local-lab.net
Server:   50 <student@local-lab.net>... Sender ok
Client:   RCPT TO: instructor@remote-lab.net
Server:   250 <instructor@remote-lab.net>... Recipient ok
Client:   DATA
Server:   354 Enter mail, end with "." on a line by itself
Client:   Received: (from student@local-lab.net)
                by mailserver@local-lab.net (8.11.2/8.11.2) id g4OK3nu19221
             for instructor@remote-lab.net; Fri, 24 May 2002 23:03:49 -0400
          Date: Fri, 24 May 2002 23:03:49 -0400
          From: student <student@local-lab.net >
             Message-Id: <200205242003.g4OK3nu19221@mailserver@local-lab.net>
          To: instructor@remote-lab.net
          Subject: Internet Lab

          The dog ate my homework.
Client:   .
Server:   250 QAA01884 Message accepted for delivery
Client:   QUIT
Server:   221 mailserver@remote-lab.net closing connection
```

Figure 0.34. Exchange of SMTP messages between SMTP client and SMTP server.

An email message may traverse multiple SMTP servers before reaching its destination. For example, on most networks a single host is dedicated to handle all outgoing email messages for all hosts of the network. In this case, all hosts forward their outgoing email message to the dedicated mail server which relays the emails to the proper destinations. Also, in many networks, the receiving mail server does not have access to the receiver's mailbox, and relays incoming messages to the mail server on a host where the receiver's mailbox resides. When a mail server relays an email message, it adds lines to the header of an email message. These lines can be used by the receiver of an email message to trace the path of an incoming email.

A mail servers adds an incoming emails to the mailbox of a user, and assumes that the user has access to the mailbox. Often, however, a user is not permanently connected to its mail server. In such situations, the user can employ mail access protocols to retrieve mail messages from their mailboxes at a remote host. Currently, two popular mail access protocols are in wide use: Post Office Protocol Version 3 (POP3) and Internet Mail Access Protocol (IMAP). Mail access protocols are generally integrated as a component of the user agent.

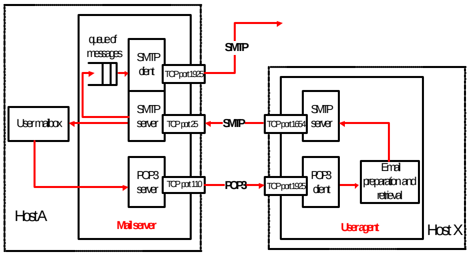


Figure 0.35. A POP3 client and a mail server.

We briefly describe the operation for a user agent that uses POP3 in Figure 0.35, where a user on Host X accesses its mailbox on Host A via POP3. The user must have a user name and a password to access Host A, and the user agent at Host X must know the domain name or IP address of Host A. The POP3 client at Host X initiates a TCP connection to the well-known TCP port 110 of the POP3 server of Host A. When the TCP connection is established, the user authenticates itself with a user name and a password. If the authorization is successful, the POP3 client can download email messages from the mailbox at Host A to Host X. The messages can be downloaded with or without deleting the message from the remote mailbox.

```
Server:   +OK POP3 mailserver.local-lab.net v7.62 server ready
Client:   user student
Server:   +OK User name accepted, password please
Client:   pass MySeCreT
Server:   +OK Mailbox open, 4 messages
Client:   list
Server:   +OK Mailbox scan listing follows
          1 861
          2 1045
          3 1560
          4 753
          .
Client:   retr 1
Server:   +OK 861 octets
          // The email follows
Client:   dele 1
Server:   +OK Message deleted
Client:   quit
Server:   +OK Sayonara
```

Figure 0.36. Exchange of POP3 messages between a POP3 client and a POP3 server.

An exchange of POP3 commands between a POP3 client and a POP3 server is shown in Figure 0.36. All commands and messages are in plain ASCII text, and lines are separated by an end-of-line sequence (CR and LF). The POP3 server acknowledges each command from the client. When the TCP connection is established, the POP3 server sends a greeting. Then the user submits a user name and a password. In Figure 0.36, the password is transmitted without encryption, but POP3 also provides more secure methods of user authentication. Once the POP3 client has authenticated itself, it issues a command to list all messages in its mailbox. The POP3 server responds with the number and length of each message. In the example in Figure 0.36, the POP3 client downloads the first message ("retr 1") and then deletes the message ("dele 1"). The POP3 client ends the POP3 session with the command "quit".

For outgoing messages, the user agent at Host X still uses SMTP. This is illustrated in Figure 0.36, where the SMTP client of the user agent at Host X connects to the SMTP server of Host A. Here, Host A acts as a relay and forwards the message to the mail server of the email receiver. Alternatively, the SMTP client at Host X can directly connect to the mail server of the receiver.

5.4. The Web

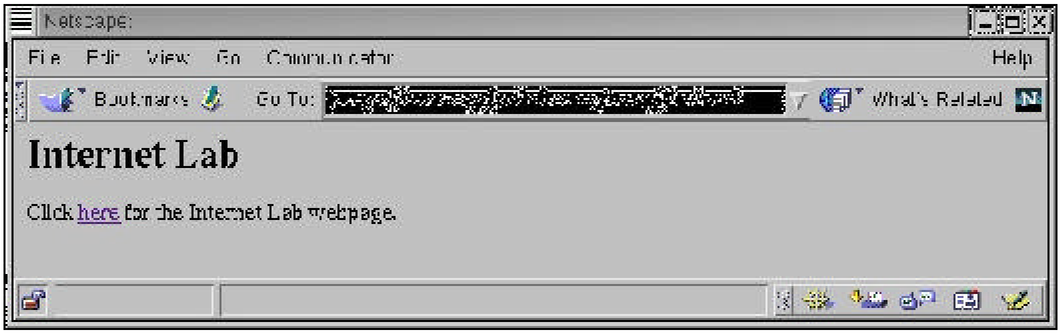
The World Wide Web (WWW or Web) emerged in the early 1990s as a new application for access to content stored on Internet hosts. Within a few years, the Web became the most popular Internet applications, and traffic on the Internet was dominated by Web applications.

The Web is a distributed hypertext system which is implemented as a client-server application. A Web client program, called a Web browser, retrieves and displays documents from a Web server. The documents, often referred to as web pages, are formatted using the Hypertext Markup Language (HTML). HTML documents are text files that contain HTML tags, which describe how text should be displayed in the user interface of a Web browser. A hyperlink is special type of tag. It is a reference to another document, which may be located at a different Web server. When displayed in a browser, hyperlinks can be activated with a mouse click. When activated, the browser retrieves the document that is referenced in the hyperlink. A Web browser makes it convenient to access a variety of documents at different web servers, which refer to each other by hyperlinks, thus, providing users with a feeling of navigating a global database of documents.

On the Web, the location of a document is expressed in terms of a Uniform Resource Locator (URL). A URL specifies a unique location for a document on the web. It can reference an HTML document, but also any other file that can be accessed with a Web browser. An example of a URL is `http://www.cerf.net/index.html`, which specifies that an HTML document with name *index.html* can be accessed via the protocol *HTTP* from a host with the name *www.cerf.net*.

```
<HTML>
 <BODY>
  <H1>Internet Lab</H1>
  Click <a href="http://www.tcpip-lab.net/index.html">here</a>
  for the Internet Lab webpage.
 </BODY>
</HTML>
```

(a) HTML file.



Internet Lab

Click here for the Internet Lab webpage.

(b) HTML file displayed in a web browser.

Figure 0.37. A minimal HTML document and its appearance in a web browser.

An HTML document and its representation in a Web browser for a minimal web page are shown in Figure 0.37. Figure 0.37(a) depicts a simple HTML document. Tags in the document are the portions of text which are enclosed by two brackets (< and >). The tag

```
<a href="http://www.cerf.net/index.html">here</a>
```

is a hyperlink which contains a URL. Figure 0.37(b) shows how the HTML document appears when it is displayed in a Web browser. The hyperlink is shown as underlined text. When a user activates this hyperlink, the Web browser attempts to access the web page that is referenced in the HTML document.

The application-layer protocol of the Web is the Hypertext Transfer Protocol (HTTP). HTTP is a request-response protocol where an HTTP client sends requests to an HTTP server, and the HTTP server responds with status information, possibly followed by an HTML file. HTTP employs TCP for data transfer, and the HTTP server uses the well-known TCP port 80 to accept requests for TCP connections. HTTP is a stateless protocol, in the sense that the HTTP server does not keep state information about request from clients. Each HTTP server handles each request from a client independently, even if multiple subsequent requests are made by the same client. There is no notion of a sessions, as, for example, in Telnet and FTP.

In older versions of HTTP, which are still in use today, an HTTP client initiates one TCP connection for each request to the HTTP server. When a single client issues multiple requests to the HTTP server, the number of TCP connections between the HTTP client and HTTP server can grow large. To reduce the number of TCP connections, HTTP/1.1, the current version of HTTP, permits multiple HTTP requests and responses on the same TCP connection, leaves the TCP connection open after a request has been served. The HTTP client does not need to wait until a request is completed before issuing new requests.

As in many other application layer protocols of the Internet, HTTP messages are transmitted as ASCII text, using the end-of-line character sequence to indicate the end of a message. The most common HTTP messages sent by a client are requests for HTML or other documents. The server responds to such a request either with a message that contains the requested document or, if the request cannot be satisfied, with an error code.

Let us know discuss a request and a response between an HTTP client and an HTTP. Suppose a user has typed the URL `http://www.cerf.net/example.html` in a web browser (see Figure 0.37(b)). When the URL is typed, the web browser starts an HTTP client which establishes a TCP connection to port 80 of the HTTP server of host www.cerf.net. When the TCP connection is established, the HTTP client sends the HTTP request shown in Figure 0.38. The HTTP request consists of the request line and a set of header line. Each line is ended with an end-of-line sequence. The request line "GET /example.html HTTP/1.1" specifies the type of command (GET), an identification of the requested resource (/example.html), and the version of the HTTP protocol (HTTP/1.1). The header lines following the request line provide additional information to the server. The request in the figure specifies the media formats, encoding schemes, and language that can be interpreted by the client (Accept: image/gif, */*, Accept-Language: en-us, Accept-Encoding: gzip, deflate), an identification of the type of web browser that issued the request (User-Agent: Mozilla/4.0), the IP address of the client (Host: 192.168.123.144), and a request to keep the TCP connection in place after the HTTP request is served (Connection: Keep-Alive). The HTTP request is completed with an empty line.

```
Client:     GET /example.html HTTP/1.1
                Accept: image/gif, */*
                Accept-Language: en-us
                Accept-Encoding: gzip, deflate
                User-Agent: Mozilla/4.0
                Host: 192.168.123.144
                Connection: Keep-Alive



Server:     HTTP/1.1 200 OK
                Date: Sat, 25 May 2002 21:10:32 GMT
                Server: Apache/1.3.19 (Unix)
                Last-Modified: Sat, 25 May 2002 20:51:33 GMT
                ETag: "56497-51-3ceff955"
                Accept-Ranges: bytes
                Content-Length: 81
                Keep-Alive: timeout=15, max=100
                Connection: Keep-Alive
                Content-Type: text/html

            <HTML>
            <BODY>
            <H1>Internet Lab</H1>
            Click <a href="http://www.tcpip-lab.net/index.html">here</a>
            for the Internet Lab webpage.
            </BODY>
            </HTML>
```

Figure 0.38. Exchange of HTTP messages between an HTTP client and an HTTP server.

The HTTP response message to the request consists of a status line, a set of header lines, and an entity body. The status line is `HTTP/1.1 200 OK` and has three parts: the HTTP version, a status code, and a status phrase. The version (HTTP/1.1) is the same as in the HTTP request. The status code is a three-digit number that states whether the request is successful, or indicates if a problem has been encountered. The status phrase provides a textual explanation to the status code. Here, the response “200 OK” indicates that the request is successful. As shown in Figure 0.38, the HTTP response has many header lines, including the date and time of the web access (`Date: Sat, 25 May 2002 21:10:32 GMT`), the type and version of the HTTP server (Server: Apache/1.3.19 `(Unix)`), the date when the downloaded document was last modified (`Last-Modified: Sat, 25 May 2002 20:51:33 GMT`), a tag that can be used to distinguish different versions of the same resource (`ETag: "56497-51-3ceff955"`), information if the downloaded file has been fragmented (`Accept-Ranges: bytes`), the length of the downloaded document in bytes (`Content-Length: 81`), a confirmation that the TCP connection will be kept open (`Keep-Alive: timeout=15, max=100, Connection: Keep-Alive`), and the media type that follows the header (`Content-Type: text/html`). Then, followed by an empty line, the HTTP server includes the requested HTML file in the entity body.

We note that the example is quite simplistic. HTTP has a significant level of sophistication, and web browsers and servers are very complex software systems. There are numerous additions and enhancements to the web, such as web caches, dynamic content, remote execution of programs at a web server, and security of transactions.

The discussion of Internet applications showed that many clients of application layer protocols set up a TCP connection to a server and then exchange protocol messages over the TCP connection as ASCII text. Since Telnet clients send single ASCII characters over a TCP connection, a Telnet client program can be used to emulate the client portion of many application layer-protocols.

Generally, a Telnet client program is started with the command

```
      telnet hostname
```

, where hostname is the domain name (or IP address) of a host on the Internet. By default, a Telnet client program attempts to establish a TCP connection to TCP port 23 of the host, which is the well-known port of a Telnet server. However, if the Telnet client is started with the command

```
      telnet hostname port
```

, where port is a port number, the Telnet client program tries to set up a TCP connection to the given port number. Thus, by setting up TCP connections to a well-known port of application-layer protocols (e.g., 21, 25, 80, 110), a user can directly interact with the servers of application-layer protocols at remote hosts. As an example, by typing

```
      telnet www.cerf.net 80
```

a Telnet client establishes a TCP connection to a remote HTTP server. After the connection is established, typing the HTTP request

```
GET  /index.html HTTP/1.1
```

at the Telnet client results in a web access of the document at URL `www.cerf.net/index.html`. In a similar fashion, a TCP client can emulate an SMTP client to send emails to an SMTP server or a POP3 client to retrieve emails from a POP3 server (Emulating an FTP client generally fails since FTP uses two TCP connections).

5.5. Recent Applications

New Internet applications are constantly emerging. In recent years, multimedia applications have emerged for the Internet that exploit the capability of modern computer systems to process audio and video data. For example, streaming media applications can retrieve and play back audio and video data, without creating a local copy of the data on the retrieving host. Application layer protocols in support of streaming media applications include RTP (Real-time transport protocol), which is used for encapsulation of multimedia data, and RTSP (Real-time streaming protocol), which supports client control functions for interactions with a streaming server. Another multimedia application is IP telephony, which refers to the technologies to support telephony in the Internet. Voice-over-IP (VoIP) is an ambitious standardization effort of IP telephony, which includes application layer protocols for signaling, for transmission of digitized voice, and for interfacing to the legacy telephone network. Peer-to-peer (P2P) applications are a new class of applications that abandon the client-server model which characterizes most Internet applications. Instead of clients and servers, there are only P2P clients. Groups of P2P clients form a virtual network and use the virtual network to exchange data.

ASCII Table

Decimal	Hexadecimal	Character	Decimal	Hexadecimal	Character	Decimal	Hexadecimal	Character	Decimal	Hexadecimal	Character
0	00	NUL (Null)	32	20	Space	64	40	@	96	60	`
1	01	SOH (Start of heading)	33	21	!	65	41	A	97	61	a
2	02	STX (Start of text)	34	22	"	66	42	B	98	62	b
3	03	ETX (End of text)	35	23	#	67	43	C	99	63	c
4	04	EOT (End of transmission)	36	24	$	68	44	D	100	64	d
5	05	ENQ (Enquiry)	37	25	%	69	45	E	101	65	e
6	06	ACK (Acknowledge)	38	26	&	70	46	F	102	66	f
7	07	BEL (Bell)	39	27	'	71	47	G	103	67	g
8	08	BS (Backspace)	40	28	(	72	48	H	104	68	h
9	09	HT (Horizontal tab)	41	29	)	73	49	I	105	69	i
10	0A	LF (Linefeed)	42	2A	*	74	4A	J	106	6A	j
11	0B	VT (Vertical tab)	43	2B	+	75	4B	K	107	6B	k
12	0C	FF (Form feed)	44	2C	,	76	4C	L	108	6C	l
13	0D	CR (Carriage return)	45	2D	-	77	4D	M	109	6D	m
14	0E	SO (Shift out)	46	2E	.	78	4E	N	110	6E	n
15	0F	SI (Shift in)	47	2F	/	79	4F	O	111	6F	o
16	10	DLE (Data link escape)	48	30	0	80	50	P	112	70	p
17	11	DC1 (Device Control 1)	49	31	1	81	51	Q	113	71	q
18	12	DC2 (Device Control 2)	50	32	2	82	52	R	114	72	r
19	13	DC3 (Device Control 3)	51	33	3	83	53	S	115	73	s
20	14	DC4 (Device Control 4)	52	34	4	84	54	T	116	74	t
21	15	NAK (Negative Acknowledge)	53	35	5	85	55	U	117	75	u
22	16	SYN (Synchronous idle)	54	36	6	86	56	V	118	76	v
23	17	ETB (End transmission blocks)	55	37	7	87	57	W	119	77	w
24	18	CAN (Cancel)	56	38	8	88	58	X	120	78	x
25	19	EM (End of medium)	57	39	9	89	59	Y	121	79	y
26	1A	SUB (substitute)	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC (Escape)	59	3B	;	91	5B	[	123	7B	{
28	1C	FS (File separator)	60	3C	<	92	5C	\	124	7C	|
29	1D	GS (Group separator)	61	3D	=	93	5D	]	125	7D	}
30	1E	RS (Record separator)	62	3E	>	94	5E	^	126	7E	~
31	1F	US (Unit separator)	63	3F	?	95	5F	_	127	7F	DEL

0.55