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

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.*

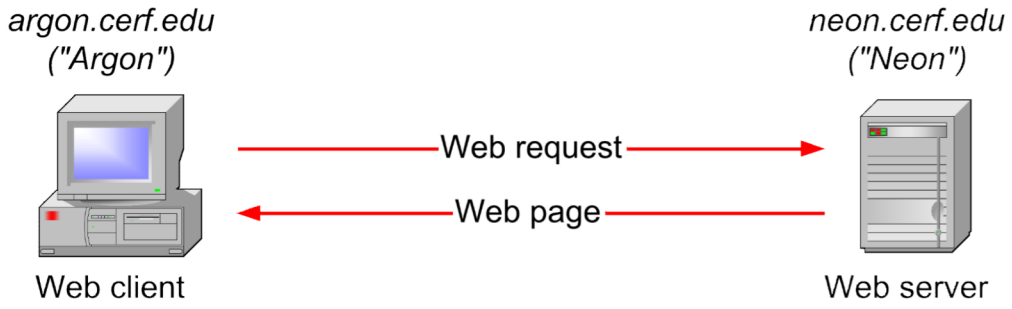

Figure 0.1. A simple web request.

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 addres “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 E thernet 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 deliv er 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 MA 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 th 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 passe 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.

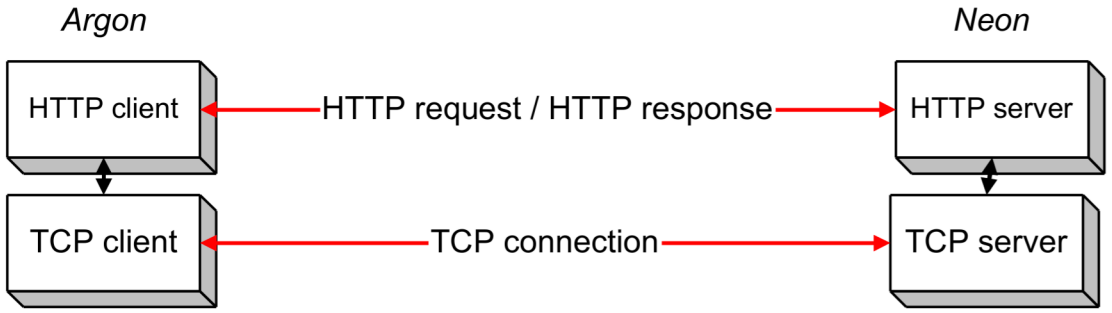

Figure 0.2 An HTTP session invoking a TCP/IP transmission

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.

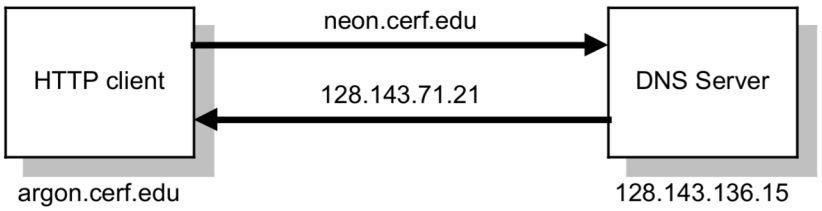

Figure 0.3. A DNS lookup for the IP address

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*.

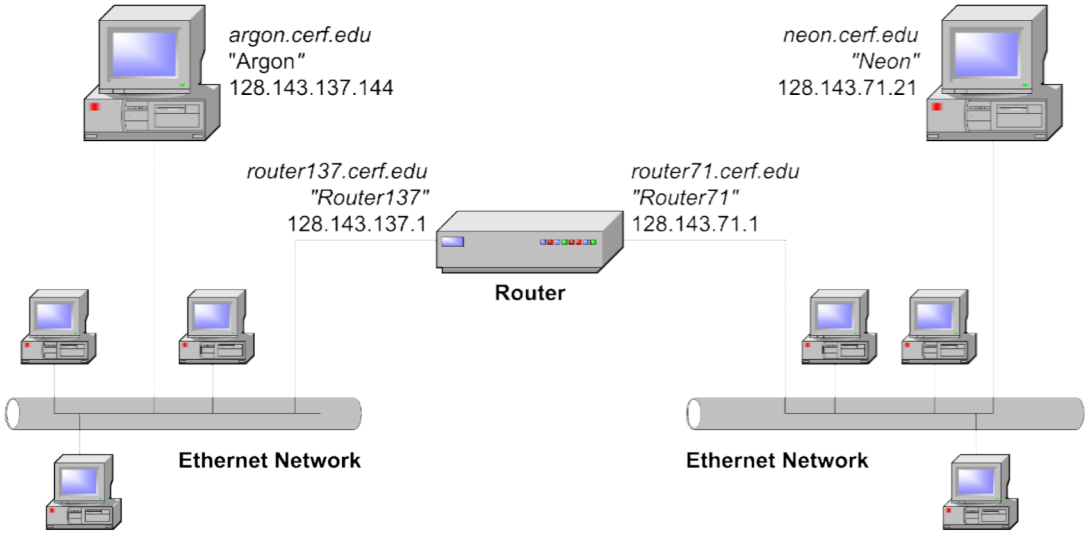

Figure 0.4. The route between *Argon* and *Neon*. The router in the center of the figure is referred to by any of the names associated its interfaces ("*Router137*" or "*Router71*").

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.)

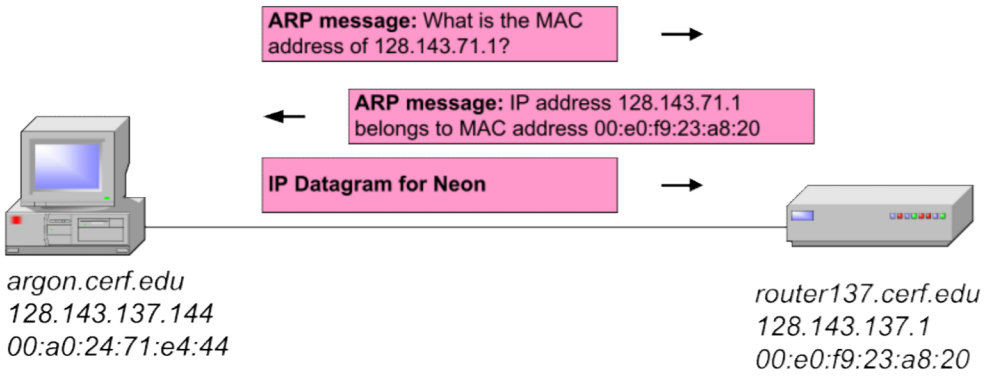

Figure 0.5. Ethernet frames transmitted to deliver the IP datagram to *router137*.cerf.edu.

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 any 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.

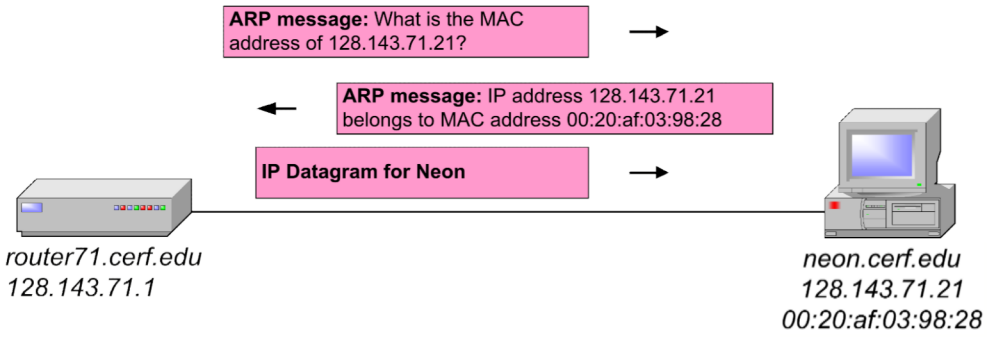

Figure 0.6. Ethernet frames transmitted to deliver the IP datagram from the router to *Neon*.

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.

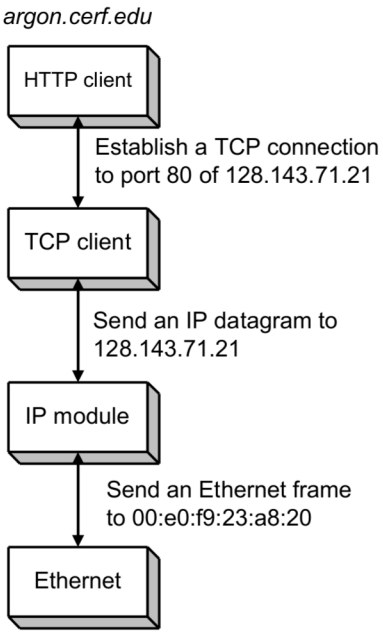

Figure 0.7. Invoking an IP connection to the remote host at IP address: 128.143.71.21

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 hardware 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.

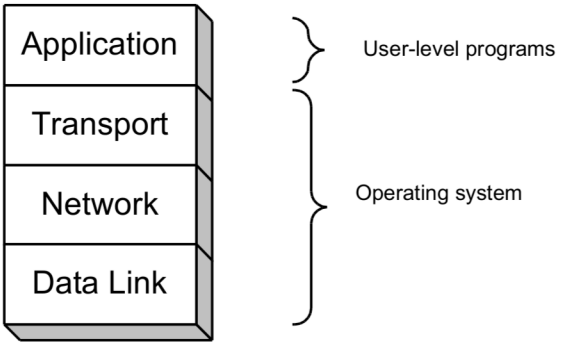

Figure 0.8. The Internet Protocol Stack

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 called *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.

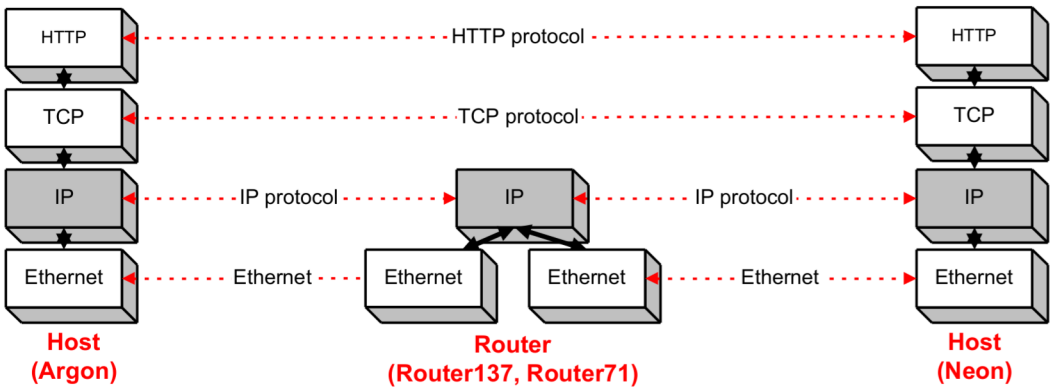

Figure 0.9. Protocols involved in data exchange between two hosts.

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.

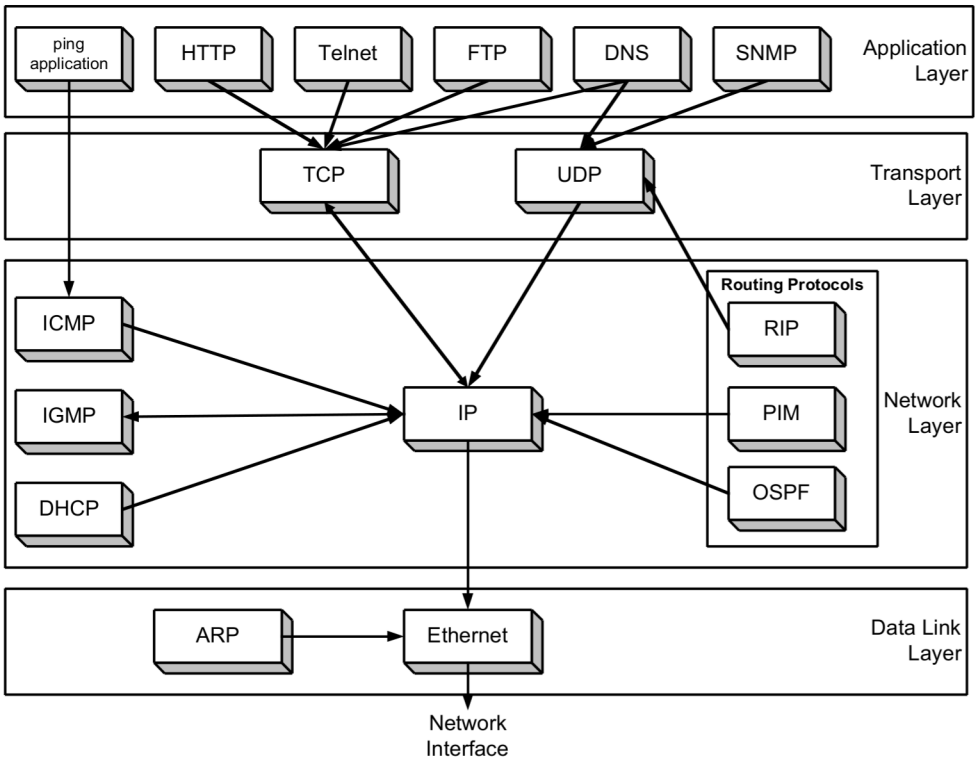

Figure 0.10. Assignment of Protocols to Layers.

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.

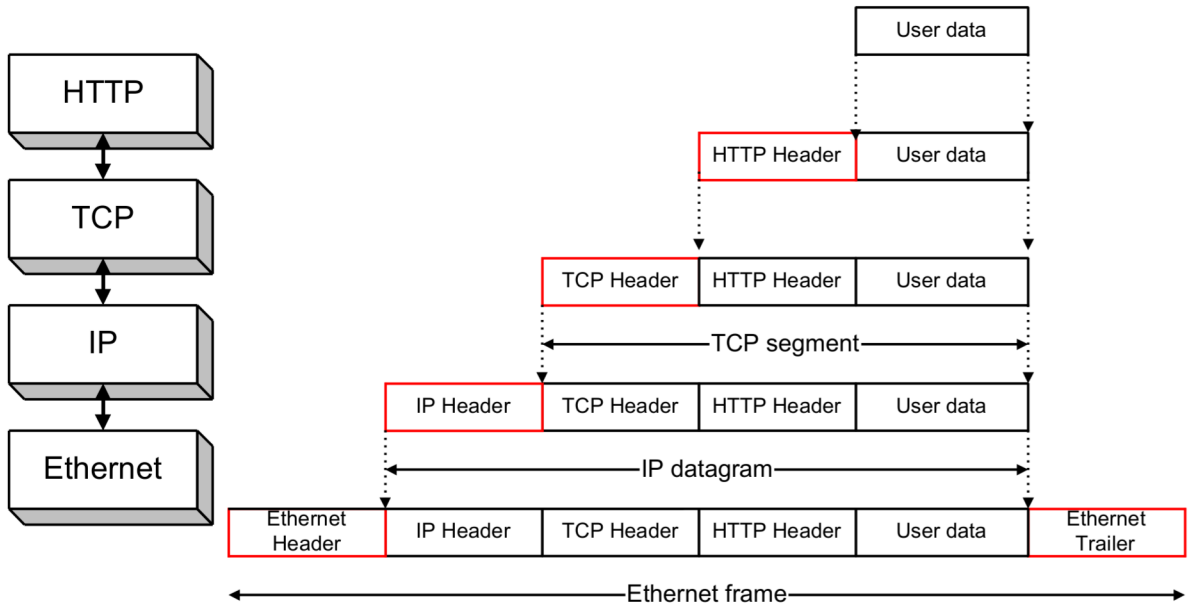

Figure 0.11. Encapsulation in the Internet.

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.

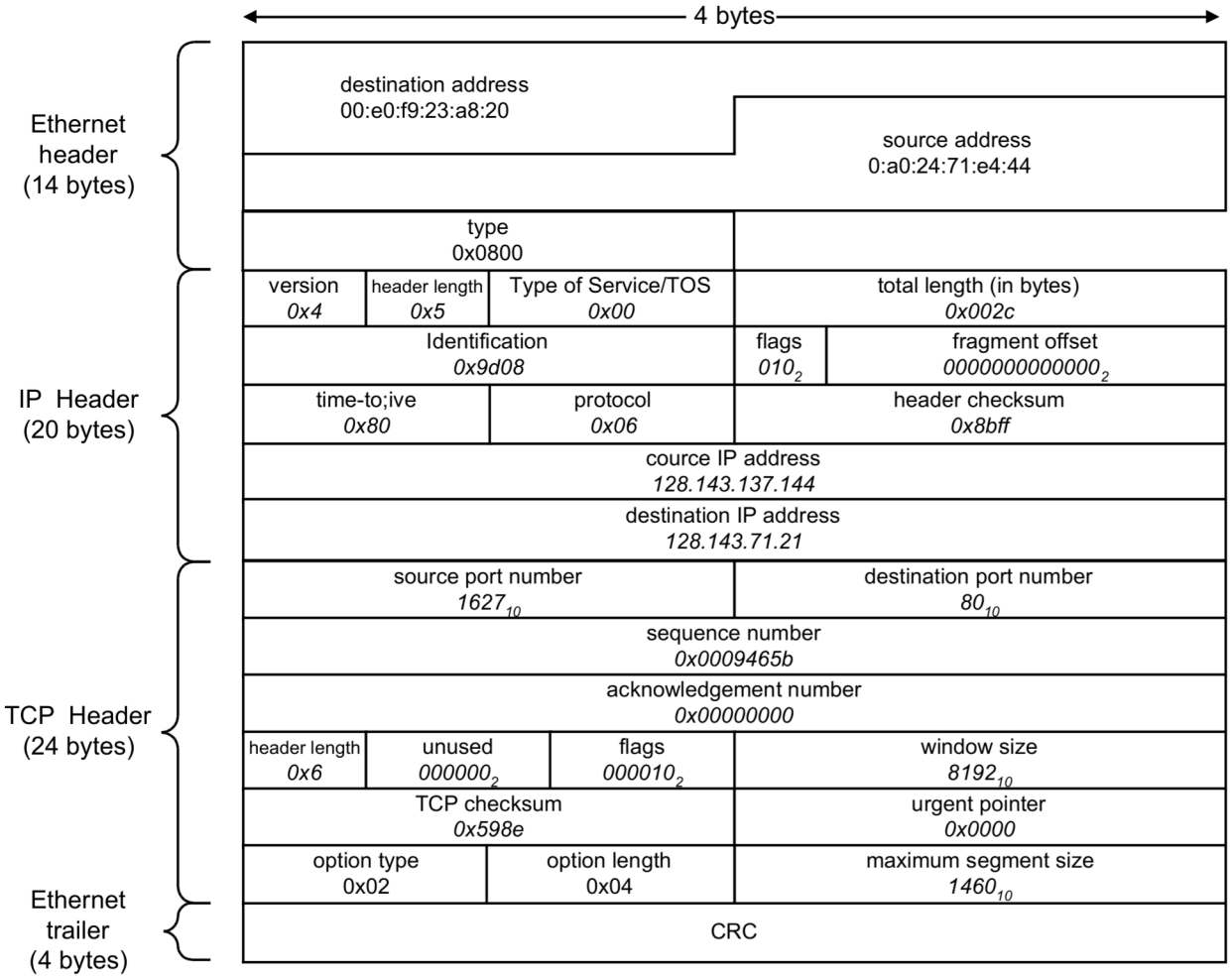

Figure 0.12. Encapsulation of a frame (0xN indicates a hexadecimal representation, N_{10} indicates a decimal representation, and N_2 indicates a binary representation of a number).

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.

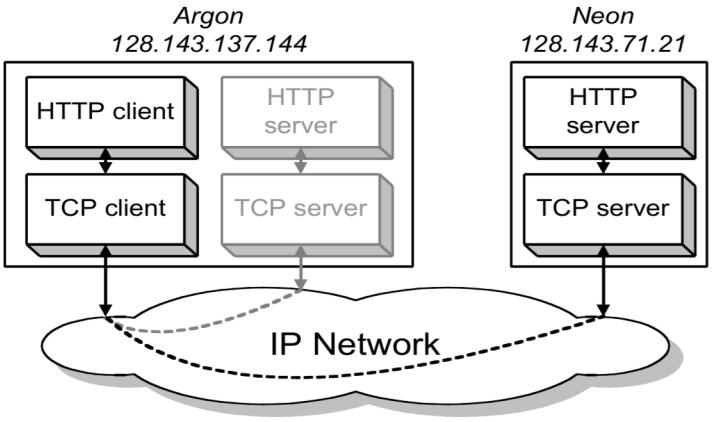

Figure 0.13 HTTP's and TCP's view of the "network"

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.

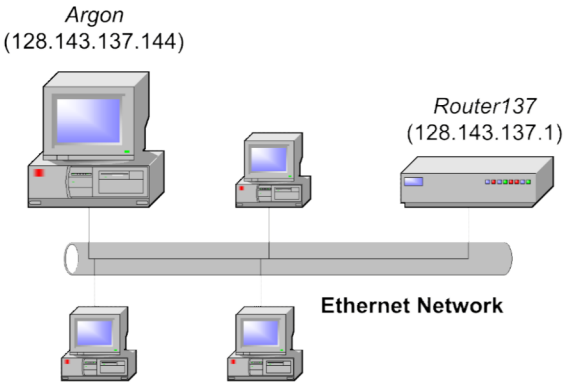

Figure 0.15. Ethernet's view of a "network".

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.

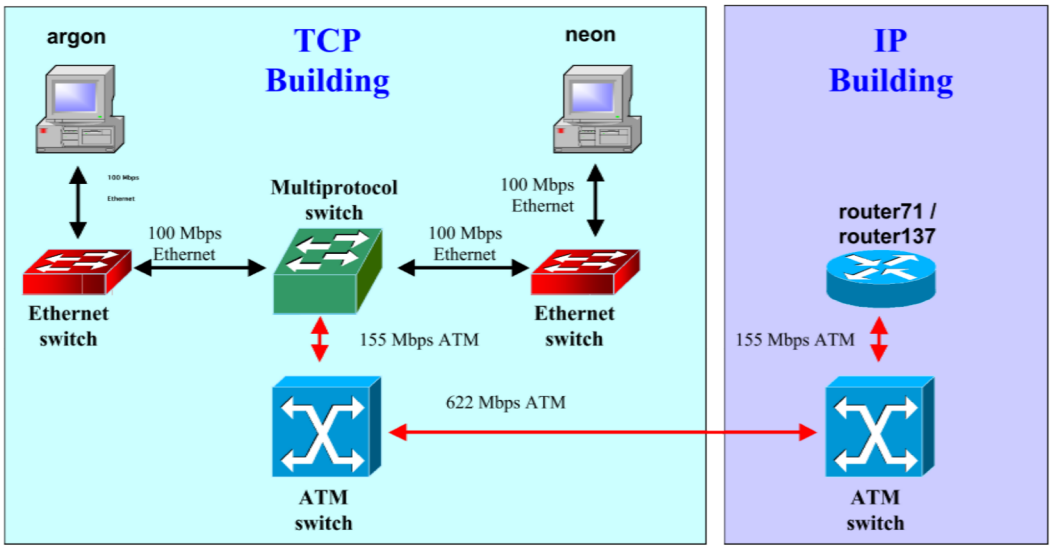

Figure 0.16. Network equipment.

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.

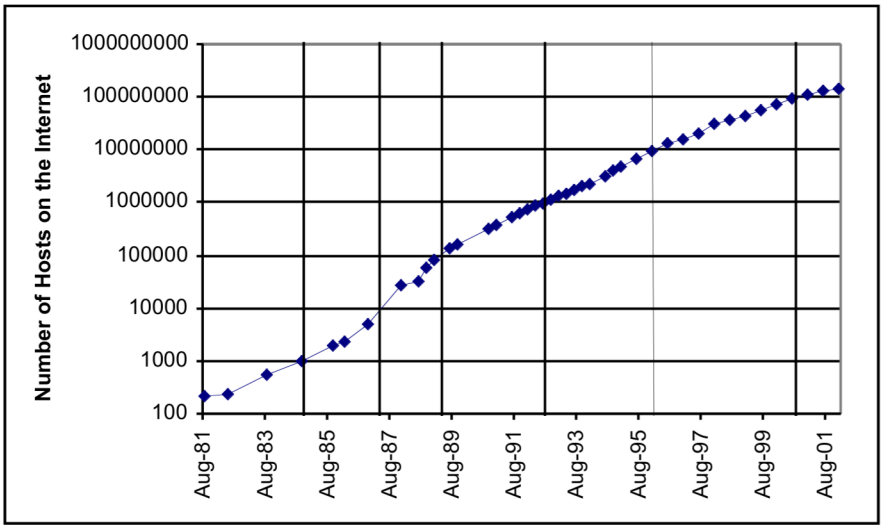

Figure 0.18. Growth of the Internet since the 1980s. The data is based on the Internet Domain Survey by the Internet Software Consortium

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.

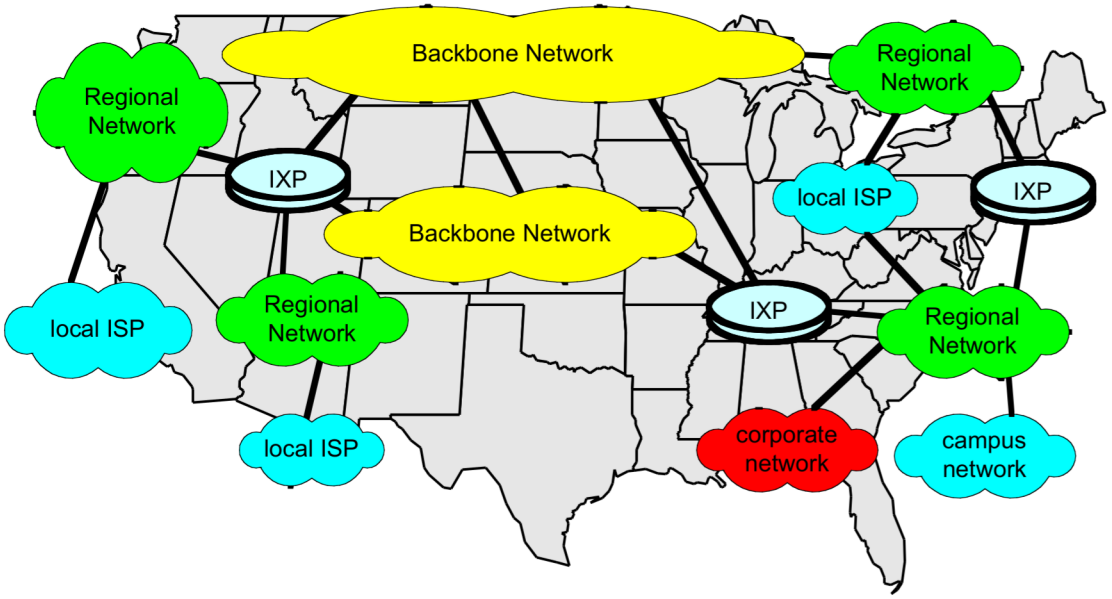

Figure 0.19. The organization of the Internet infrastructure.

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.