TCP/IP

INTRODUCTION

The internet uses a variety of technologies to provide end-to-end communication between applications on different computers. Most applications use Transport Control Protocol (TCP) and Internet Protocol (IP). Some applications use User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP). All of these protocols use local network technologies such as ethernet and dial up modems. This lesson will explain many of the technical aspects of these communication technologies.

PREREQUISITES

Home Networking

TRANSMISSION CONTROL PROTOCOL (TCP)

TCP provides reliable, in-order delivery between two computers. TCP is a connected protocol, which means that TCP establishes a link between two computers, similar to the way that a telephone establishes a link between two people. Once the connection is established, any data that's put in one end of the connection comes out the other end of the connection. Like a telephone, TCP connections are full-duplex, meaning that both computers can send data and both will receive data sent by the other computer.

TCP is a packet data protocol. This means that the TCP module breaks incoming data into small chunks, called packets (actually, TCP packets are called “segments”). The maximum size of a TCP segment can be adjusted by the TCP module, but must be less than 65536 bytes. The TCP module adds a 20 byte header to the beginning of each segment (see Figure 1), then passes this TCP segment to the IP module for transmission.

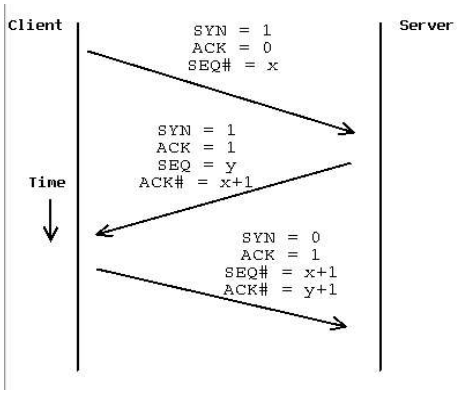

Figure 2: TCP Connection

Before an application can communicate using TCP, the TCP module must create a connection. First, TCP creates a socket on each computer. A socket is the end point of a TCP connection and consists of an IP address and a port number. Generally, port numbers correspond to applications or services – the **`/etc/services`** file lists valid services and their port numbers. Typically, a server application, such as a web server, will instruct the TCP module to create a listening socket. A listening socket waits for another computer to contact it. A client application, such as a web browser, creates a socket and instructs the TCP module to connect its socket to the listening socket on the server.

Once the sockets have been created, TCP uses a three way handshake mechanism to establish the connection, as shown in Figure 2. First the client socket sends a TCP segment to the target socket with SYN set to 1, ACK set to 0, and some number (say “x”) in the sequence field. If the target socket is listening and intends to accept the connection, it replies to the originating socket with a TCP segment containing SYN set to 1, ACK set to 1, Acknowledgement number set to x+1, and sequence number set to some number (say “y”). Finally, the client should confirm the connection with a TCP segment

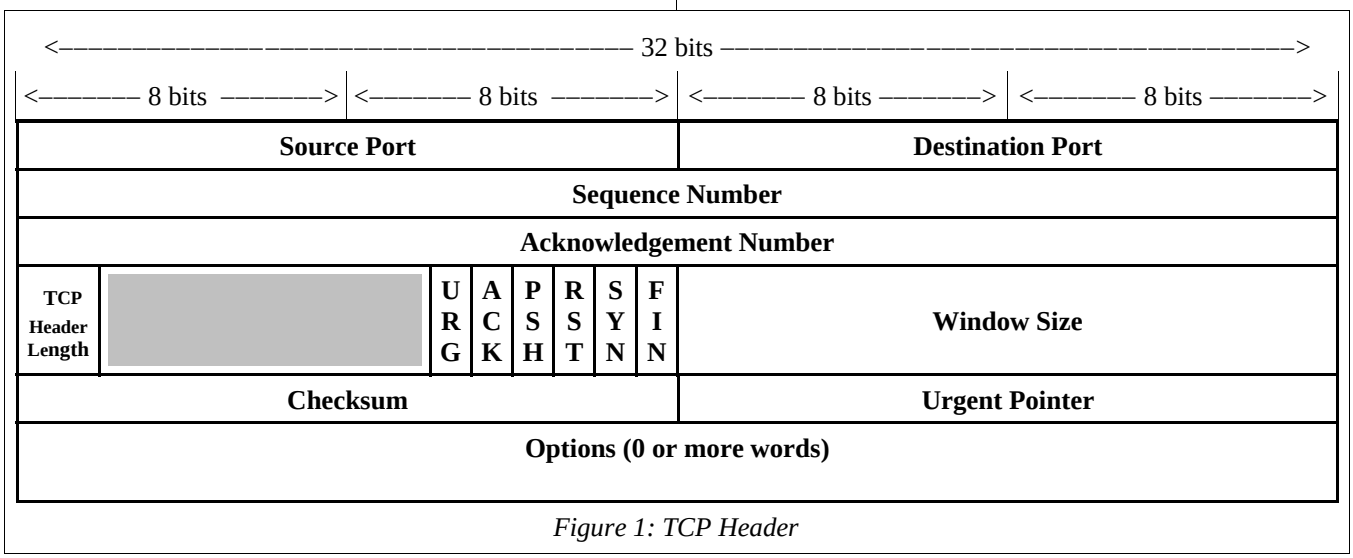

Figure 1: TCP Header

containing SYN=0, ACK=1, ACK#=y+1 and SEQ#=x+1. The connection is now established and the two computers may exchange data on the socket.

Port Scanning & Syn Floods

This process might seem a bit tedious, but it is very important from a firewall point of view because bad guys misuse SYN segments. The black hats will scan all the target sockets at a given IP address by sending a SYN=1 ACK=0 segment to every port. If the target computer replies with a SYN=1 ACK=1 segment on that port, then the bad guy knows that there is a socket listening on the port. But they will never send the SYN=0 ACK=1 packet, so the connection is never opened, and often goes unlogged at the target. This is sometimes called “ghost scanning” - bad guys can learn about your computer without you knowing that they are there.

Once the attacker has identified a listening socket he can try a more serious attack - he sends a very large number of SYN=0 ACK=1 packets from a forged IP address to a listening port on the target computer. By “forged IP address” I mean that the bad guy has manipulated his IP module to insert an incorrect value in the source address field in his IP headers. Each time the listening socket receives a SYN=0 ACK=1 segment, the listening socket allocates resources for a new connection and transmits a SYN=1 ACK=1 segment. But the bad guy never replies. When it doesn't receive a reply, the server will keep its end of the connection open and transmit another SYN=1 ACK=1 segment. These half open connections are called “half open connections”. Each half open connection consumes resources (RAM, stack space, CPU cycles) on the server; when the server uses up too much resources it denies new connections, slows down or crashes. This type of attack is called a SYN flood. It is very difficult to trace the attacker because he never uses his real IP address.

User Datagram Protocol

There may be times when an application doesn't need a connection based protocol – maybe the application just needs to send ten or twenty bytes of data, and doesn't need any guarantee of delivery. For small messages that don't require guarantee of delivery, internet applications use the User Datagram Protocol (UDP). Like TCP, UDP sends messages from one socket to another. Unlike TCP, UDP does not create a connection between the two sockets. Since it doesn't create a connection the UDP module cannot segment application data into smaller packets – the application must deliver data to the UDP module in small chunks that will fit in a single datagram. Typically datagrams are less than 100 bytes long, but they might be as large as 65535 bytes (including the 8 byte header). Also, because there is no connection to maintain, the UDP header doesn't need to put as much information in the header. The UDP header is only 8 bytes long as depicted in Figure 3.

Although UDP and TCP both use ports, the assignment of services to ports is different for the two protocols. The `/etc/services` file lists services for both UDP and TCP. Some of the more prominent applications which use UDP are:

- trivial file transfer protocol, UDP port 69
- windows networking (netbios), UDP ports 137, 138, and 139
- name server, UDP port 42
- network time protocol, UDP port 37

Internet Control Message Protocol

Occasionally internet gateways may need to send messages to source computers – perhaps the destination is unreachable, or a header field has been corrupted. Gateways communicate with source computers using the Internet Control Message Protocol (ICMP). ICMP is defined in RFC 792. ICMP is a connectionless protocl – it uses datagrams. ICMP messages are used by the IP module, not by applications or services, so ICMP does not use sockets. ICMP datagrams are small: a header of 8 to 20 bytes, followed by no more than 84 bytes of data. There are six types of ICMP datagrams. The exact format of the datagram depends on the type, but the first byte of the datagram always identifies the ICMP type. The types are:

- type 3: Destination Unreachable. The destination source computer cannot be reached on the network, or the service/protocol is not available on the destination computer.
- Type 4. Source Quench. Packets are arriving too fast and the gateway does not have enough buffer space to store arriving packets.
- type 11: Time Exceeded. The Time To Live value in the IP header has reached zero or an internet node was unable to reassemble a fragmented packet. Time to live is described in RFC 791. We will discuss fragmentation later in this lesson.

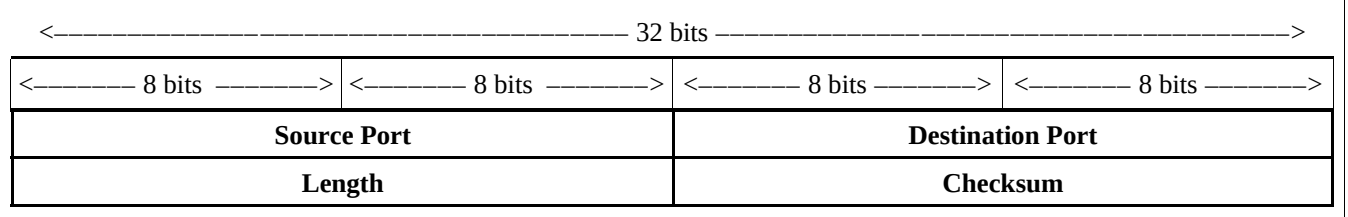

Figure 3: UDP Header

- type 12: Parameter Problem. A gateway or host cannot process the packet because of an error in the header parameters.
- Type 5: Redirect. The packet was sent to the wrong gateway. A type 5 message identifies the correct gateway.
- Type 8: Echo. A gateway or host requests an echo reply. Used by the ping program.
- Type 0: Echo Reply. Reply to a ICMP type 8 datagram.
- Type 13: Timestamp. A gateway or host requests a timestamp reply.
- Type 14: Timestamp reply. Reply to an ICMP type 13 datagram.

INTERNET PROTOCOL

TCP, UDP and ICMP use the Internet Protocol (IP). IP moves data across the network. IP provides no guarantees of service – packets might arrive at their destination late, out of order, or may not arrive at all. IP is said to provide "best effort" delivery. If an application requires better quality assurance, then it is the job of the application to provide the assurance or to use a higher level protocol, such as TCP, which provides some guarantees.

IP is a packet protocol. The IP modules segments incoming data into chunks and adds a header to each chunk to create an IP packet. The maximum size of an IP packet is 65535 bytes, and the IP header is 20 bytes long, so each data chunk can be no more than 65515 bytes long. But in practice the data size is much smaller (we will discuss the Maximum Transfer Unit value later in this lesson).

Figure 4 shows the layout of an IP header. RFC 791 describes all the fields – we will only discuss a few. The most important fields are the source address and destination address. We discussed IP addresses in the Home Networking lesson. The IP addresses specify which computer sent the packet, and which computer should ultimately receive the packet. Protocol is another important field. The protocol field identifies which higher level protocol is using IP. The /etc/protocols file lists valid protocol numbers and their corresponding protocols. For instance ICMP is protocol 1, TCP is 6 and UDP is 17. You should know about the fragments field, but before we discuss fragments you must understand the idea of a Maximum Transfer Unit.

Maximum Transfer Unit

RFC 791 defines the maximum sized datagram that can be transmitted through the next network as the maximum transmission unit (MTU). Different networks use different technologies, and different technologies allow different maximum sized datagrams. For instance, the maximum size of a ethernet frame is about 1500 bytes, the maximum size of a frame relay packet data unit could be anywhere from 622 to 4096 bytes and the maximum size of an ATM cell is 56 bytes.

A single packet may traverse many different networks between its source and destination. These different links might use different network technologies having different MTU's. Consider the network depicted in figure 5. The originating computer is directly connected to an ethernet network, so it generates packets of 1492 bytes (less than the network MTU of 1500 bytes). But the second network uses frame relay technology with a PDU size of 1024 bytes. The third network is another ethernet. The problem is that when the packets of 1492 bytes arrive at the gateway to the second network, the packets are larger than the second network's MTU. There are two ways of dealing with this problem. We will talk about one of these methods: IP fragmentation.

IP Fragmentation

The second network only accepts datagrams of 1024 bytes or less. So if Gateway B receives a packet from Source A sends a packet that is larger than 1024 bytes, Gateway A must fragment that large packet into two or more smaller packets. Gateway B accomplishes this task by separating the data from the original packet into two

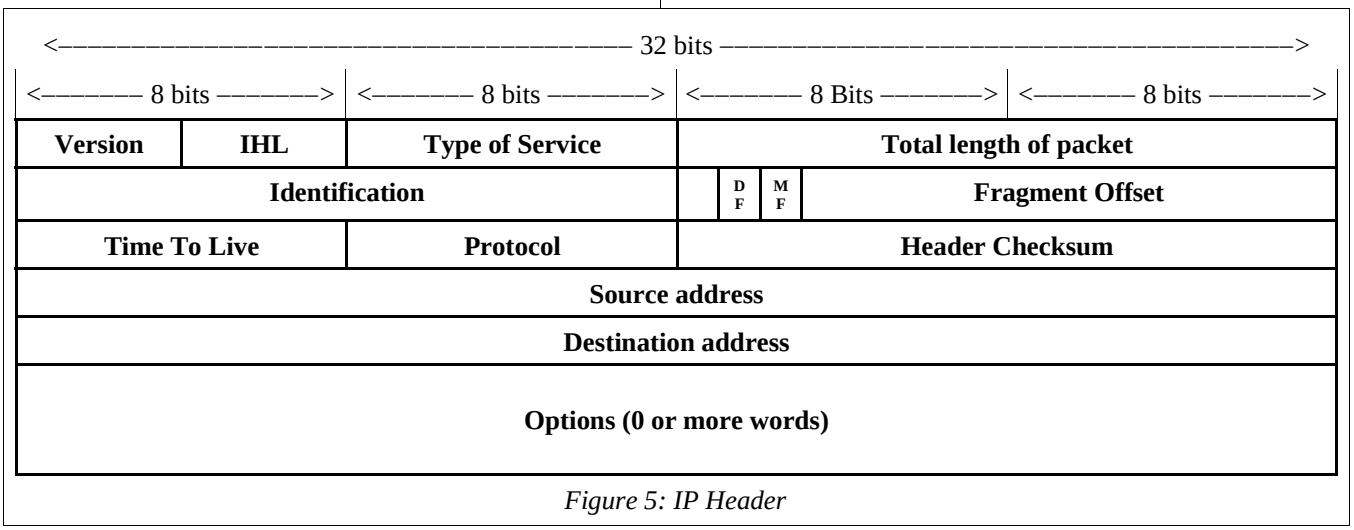

Figure 5: IP Header

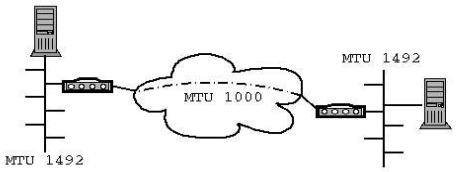

Figure 5: Fragmentation

smaller chunks, in this case suppose that the first chunk is 800 bytes and the second chunk is 672 bytes. Then the gateway copies the original IP header to each of the new chunks, creating two new IP packets. The gateway sets the fragment offset field of the first new packet to 0 and the second new packet to 1. The gateway sets the MF (More Fragments) bit of the first new packet to 1, meaning that there are more fragments following this one. The gateway sets the MF bit of the second packet to 0, meaning that this is the last packet. Then the gateway transmits both new packets on the frame relay network.

The two fragments could be reassembled into a single packet at the other gateway or at the destination computer. From an IP perspective, it doesn't matter where the reassembly is performed, because other than the fragment fields, all fragments have identical headers, so they should all follow the same path.

Don't Fragment

The Don't Fragment (DF) bit in the IP header instructs gateways and intermediary hosts not to fragment the packet. If the packet is too big to fit on the next network, the gateway drops the packet.

MTU Tuning

Consider the internetwork in Figure 5 one last time. Suppose that Host A's sends many packets that are more than 1024 bytes long. In order to transmit these packets over the frame relay network, the gateway splits fragments each packet. Then the destination reassembles the fragments. The process of segmenting and reassembling fragments takes time. Also, the PDUs will only be about 75% full on the frame relay network. Since each original packet results in two headers transmitted on the frame relay network, in this case fragmentation effectively increases the IP header overhead by 50%. All of these inefficiencies result in slower transmission. But these inefficiencies can be avoided by tuning the MTU at the source. The actual procedure for tuning the MTU is complicated and time consuming, but there are programs available on the internet which will automate the process.

LOCAL NETWORKS

Applications transfer data using TCP. TCP transfers segments using IP. And IP transfers packets using the “local network”. Local network is an old and outdated term. Some of the networks are not local at all: consider the national ATM networks which many telephone companies have installed. But the term is defined in RFC 791 so we will use it. We will discuss several local network technologies.

Ethernet

Without a doubt, ethernet is now the most widely deployed local network technology. The Institute of Electrical and Electronics Engineers (IEEE) defines many different types of ethernet, but they all have the same basic frame structure, defined by the IEEE 802.3 standard. Figure 6 depicts the ethernet frame structure. The seven byte preamble contains the bit patttern 10101010... which helps synchronize the sender's and receivers' clocks (ethernet is a point-multipoint technology, so there can be one sender and many receivers). The eighth byte is 10101011 which delimits the synchronization field from the rest of the frame. Ethernet includes six byte source and destination addresses. These addresses are usually called Media Access Control (MAC) addresses. There exists an option for 2 byte MAC addresses, but it is very rarely implemented. The length of the data field may be zero to 1500 bytes. But the ethernet protocol specifies that there must be at least 64 bytes from the beginning of the destination address to the end of the checksum. But the addresses, length field and checksum only add up to 18 bytes, so if there is less than 46 bytes of data, the ethernet controller will add the required number of bytes to the Pad field.

RFC 894 defines the methods for transmitting IP packets over ethernet networks.

Point to Point Protocol

Internet Standard 51 (RFC 1661) defines the Point to Point Protocol (PPP). PPP may be used to transmit many different types of datagrams, but we are only interested in IP over PPP. PPP is not really a local network – it is a simple protocol which allows easy communications over a variety of different types of networks. In a sense, PPP fits between IP and the Local Network in Figure 7. The

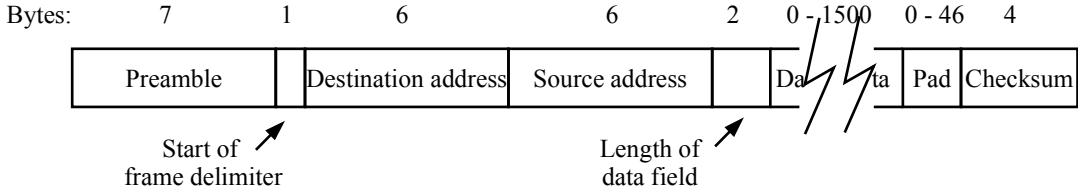

Figure 6: Ethernet Frame Structure

two limitations on these networks are that they must be point-to-point and they must be full-duplex. This means that there are only two computers on the network and data transmitted by one computer is received by the other computer and vice versa. Because PPP may be transmitted over many different types of local networks, there must be a standard to define each implementation. For instance, RFC 1662 describes the common implementation of PPP over serial cables or telephone modems. There are more esoteric applications of PPP: RFC 2516 defines PPP over ethernet (PPPoE) and RFC 2364 describes PPP over ATM (PPPoA). PPPoE is often used to provide an inexpensive, high bit rate link between a computer and an xDSL or cable modem. PPPoA can be used to establish multi-protocol data channels over ATM virtual circuits.

PUTTING IT ALL TOGETHER

Figure 7, copied from RFC 791, illustrates the idea that applications use high level protocols such as TCP and UDP; high level protocols use IP and ICMP; and IP and ICMP use the local networks. PPP fits between IP and the local network protocol.

It might also be constructive to consider what happens to a small chunk of data which is transmitted over a network using TCP/IP. Suppose an application sends a small, 30 byte message to another computer. The application sends the data to a TCP socket. The TCP module adds a 20 byte header and passes the TCP segment to the IP module. The IP module adds a 20 byte header and passes the IP datagram to the ethernet module. The ethernet module adds a 22 byte header and

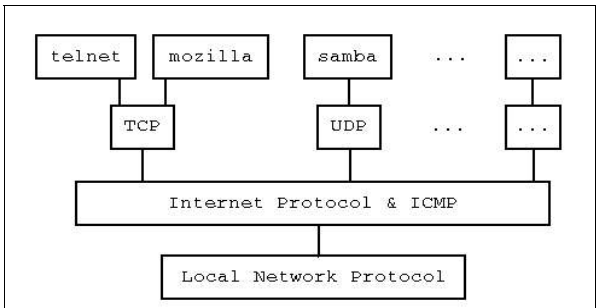

Figure 7: Protocol Relationships

a 4 byte checksum and transmits the ethernet frame. The end result, illustrated in Figure 8, is that a large portion of the transmitted packet is information about the packet. We will use this idea in the next lesson: Packet Filtering Firewall.

REFERENCES

RFC 791 Internet Protocol

RFC 792 Internet Control Message Protocol

RFC 793 Transmission Control Protocol

RFC 768 User Datagram Protocol

Computer Networks Third Edition, Andrew S. Tanenbaum, Prentice-Hall 1999

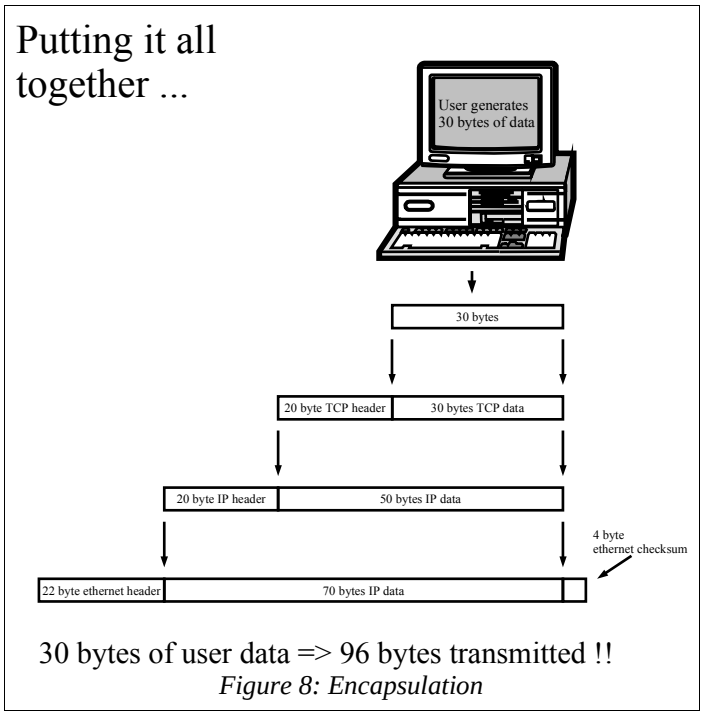

Figure 8: Encapsulation