UDP and TCP

This lecture describes the two most common transport-layer protocols used by IP networks: UDP and TCP.
After this lecture you should be able to:
calculate and explain the values of the following UDP and TCP header fields: source and destination port numbers, checksum, length, sequence number, acknowledgment, flags and window; predict the change of TCP state machine state as a result of a socket API call or receiving a frame with specific bits set; predict the maximum amount of data that will be sent in the next TCP segment based on the sequence numbers of queued data and the values in a received TCP header.

Introduction

So far we have studied the data link layer which is responsible for getting packets from one end of a transmission line to the other, and the network layer which is responsible for getting IP packets from host to another. In this lecture we look briefly at the transport layer, a set of higher level protocols that make it easier to write networking applications.

These applications include clients such as web browsers or servers such as file servers. The transport layer serves to connect client applications on one host to the desired server processes on another. In addition, some transport protocols provide other services such as reliable, in-sequence message delivery and congestion control that are not provided by the underlying network layer (IP).

There are two transport-layer protocols in common use:

UDP - sends packet from a port on one machine to a port on another machine; relatively simple

TCP - use a "connection-oriented" model that provides reliable byte pipe from a process on one machine to process on another; relatively complex

Other abstractions, such as a reliable datagram protocol, would be possible but having only two keeps things simple for implementors.

Transport Layer Functions

The most important features we would like to get from a transport protocol include:

- reliable delivery: sender receives an acknowledgment that the data was received
- sequencing: packets are guaranteed to arrive in the same order they were sent
- no duplication: data is not duplicated
- fragmentation: arbitrarily-long byte sequences can be broken down into smaller packets (called "segments" in TCP) and reassembled at the destination
- flow control: the receiver can be told (or can infer that it should) slow down the rate at which it sends data to a destination
- multiplexing: we can have multiple simultaneous connections between hosts, even to the same service

In the same way that IP addresses are used to specify a particular host, a server can provide multiple types of services. Each such service needs an "address" or "service access point" (SAP). In TCP/IP SAPs are defined by an associated 16-bit numbered "port". Various services have well-known port numbers (e.g. 80 for HTTP).

Each transport-layer IP connection is uniquely defined by the combination of 4 things: host and destination addresses and host and destination port numbers. Each different 4-tuple defines a different connection. This explains how, for example, multiple users on the same client can contact the same service on the same server.

Exercise 1: Two users on the same client connect to the same web server. Which of the addresses and ports are the same? Which are different?

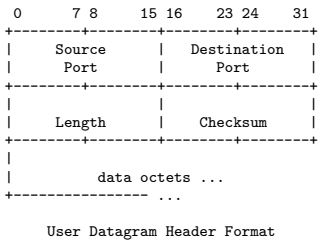
UDP

The UDP (User Datagram Protocol) is a simple protocol that provides very little more that is provided by IP. UDP is defined in RFC-768. UDP does not offer flow control, reliable delivery, un-duplicated or in-sequence delivery. Instead,these features must

be provided by the application. This allows more flexibility at cost of additional complexity in the application.

UDP Header

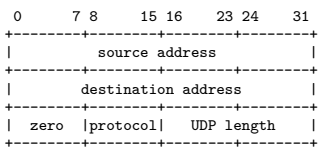
The following diagram, from RFC-768, shows the UDP header:



User Datagram Header Format

The header has four 16-bit words: source and destination ports, length, and a checksum. The main purpose of the header is to specify destination service (port, SAP) that should handle the packet. The source port is included so it can be used to address a UDP datagram as a response.

The length is the number of bytes in the datagram, include the header. The checksum is the regular IP 16-bit one's complement checksum applied to the header, data and a selected portions (3 32-bit words) of the IP header (the IP "pseudoheader") as shown below:



Exercise 2: What are the minimum and maximum values for the UDP length field?

Examples of UDP-Based Applications

A typical UDP-based application is the DNS. A single UDP packet containing the query is sent to a DNS server and a single UDP response is received with the response. The application handles duplicate or lost frames.

Another class of applications for UDP is interactive applications such as VoIP (voice over IP). Latency is very important for interactive applications – sometimes more important than avoiding loss of data. The reliable-delivery and congestion control features of TCP can sometimes result in long delays. Interactive applications often use UDP so they can implement their own error- and congestion-control algorithms.

TCP

Transmission Control Protocol (TCP) is the most commonly-used transport-layer protocol. It is defined in RFC 793.

TCP is a full duplex protocol – it provides reliable byte streams in both directions.

Like UDP, TCP connections are established between ports that allow multiple links between hosts and also allow selection of the that clients connect to.

A TCP connection has state that needs to be maintained for each connection (host/port addresses, sequence numbers, buffers, timers, ...). This state needs to be initialized when the connection is set up and the memory freed when the connection is terminated.

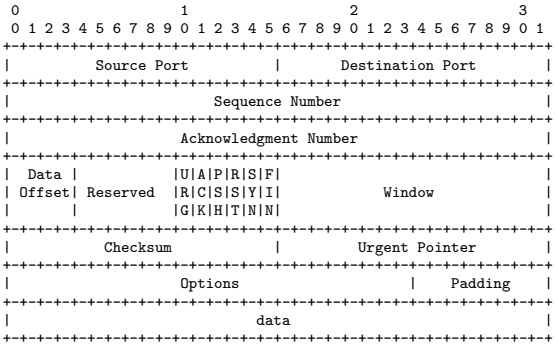
To deal with highly variable network delays and data rates, TCP uses a sliding window ARQ protocol for congestion and flow control.

It is important to understand that although the TCP frame format and the protocol state machine are defined in RFC 793, there is considerable variation in TCP implementations because:

- TCP allows for options to be negotiated at the start of the connection; hosts may implement different sets of options, and
- the algorithms used to, for example, decide how long to wait before transmitting or retransmitting a frame are implementation-dependent.

TCP Header

The following diagram, from RFC-793, shows the contents of the TCP header:



The source and destination ports have the same meaning as in UDP.

The TCP protocol chops up the data stream into "segments," each of which transmitted in separate IP frames. The Sequence Number is the sequence number of the first byte in the segment carried by this frame.

The Acknowledgment Number is the sequence number of the last byte received by the sender of this frame.

The Window value can be interpreted to mean how many bytes of buffer space the receiver currently has available.

Note that Sequence number refers to the data being transmitted in the same frame while Acknowledgment and Window refer to the connection in the opposite direction.

Exercise 3: The most recently received TCP packet for a connection had an Acknowledgement Number value of 1000 and a Window value of 64. Assuming 1024 bytes are ready to be sent, what will be the value of Sequence Number in the next packet transmitted for that connection? What will be the length of the IP packet?

The Data Offset the number of 32-bit words in the header (including options).

The 6 bits in the flag field are used to signal the state of the connection between peers and manage the TCP state machine.

The URG and ACK bits indicate that the values in the Urgent Pointer and Acknowledgment Number fields are valid.

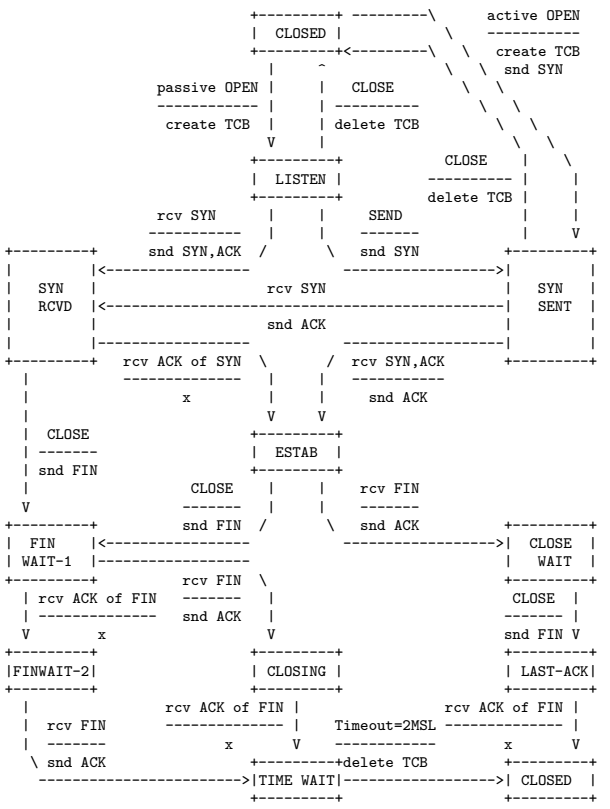
The SYN and FIN bits are used to start and terminate a connection.

The PuSH bits can be used to indicate the end of a message defined by a higher-level protocol. The RST bit is used to indicate a loss of synchronization (e.g. Acknowledgment of data not sent) and causes the connection to be closed.

The checksum is computed over the data, TCP header and an IP pseudo-header as with UDP.

TCP State Machine

The ASCII diagram below, taken from RFC-793, describes the TCP state machine. The text in each block labels that state. The labels on the state transition path state the action or received flag bits that cause the transition (above the line) and the bits that are set in the frame that is sent in response



TCP Connection State Diagram
Figure 6.

(below the line). Other labels are x (send nothing), active/passive OPEN (client/server socket creation), create/delete TCB (allocate/release data structures for the connection) and a Timeout (2×120 seconds).

Study of the diagram shows that setup and teardown of a TCP connections requires a 3-way handshake { SYN, SYN+ACK, ACK } or { FIN, FIN, ACK }.

The above is a quick summary of how connections are set up and torn down. There are many more details which we do not have time to cover in this course.

Exercise 4: What will happen if a host responds to an initial SYN frame by sending back a frame with only SYN set?

Congestion Management In TCP

When more data arrives at a router than can be sent out, the data will build up in queues. This is called congestion. If the queues get too long then packets will have to be dropped. These frames then have to be retransmitted. This increases the load on the network which can then become unstable.

Since each packet is routed independently, the network layer cannot determine packet delay or loss. On the other hand, TCP sees acknowledgments and so it can measure delays and take appropriate measures to deal with congestion. The transport layer is thus made responsible for congestion management and flow control (deciding on the average transmit rate).

One of the main purposes of TCP is to minimize the likelihood of dropped frames by reducing the rate at which frames are sent out when congestion is detected. The algorithms to do this are fairly complex and implementation-dependent. We will not cover them here.

Exercise 5: How can a host reduce the rate at which data is sent to it on a TCP connection?

Examples of TCP-Based Applications

Most internet applications use TCP. This includes web browsers (the HTTP protocol, server on port 80), e-mail transfer (the SMTP protocol, port 25), secure shell login (the SSH protocol, port 22) and many others.

Most of these protocols use text-based dialogs. For example, an an HTTP client establishes a TCP connection and then sends a line starting with GET followed by the URL and a CR character. The server response with some headers and the page contents. Similarly, the SMTP protocol uses commands formatted as lines beginning with commands such as HELO, FROM, MAIL and DATA to transfer e-mail messages.

The Socket API

The "socket" library is commonly used to implement network client and server software that uses the TCP/IP protocol. The API (Application Programming Interface) is a set of functions that can be called by a program to implement the networking parts of the program. Although you do not need to be able to write such applications, it is useful to know how the API is accessed and you should know which of these calls cause changes of state in the TCP state machine. The main functions are:

socket() - allocates and initializes the networking data structure, similar to open() for reading or writing files

bind() - sets the local address (port) (used on server side)

listen() - enables incoming connections (on server) – the "passive open" described above

accept() - waits for incoming connections (on server)

connect() - waits for outgoing connections (on client) – the "active open" described above

send() write()] - sends bytes over the connection

receive()read() - reads bytes received over the connection

close() - shuts down the connection and releases memory – the "close" described above

In addition to the functions above, other library functions are often used. For example, the `gethostbyname()` function can be used to convert a host name to an IP address. The method is implementation-dependent way, but typically it's by querying the DNS.