1.1 Transmission Control Protocol

TCP is often described as a byte stream, connection-oriented, reliable delivery transport layer protocol. In turn, we will discuss the meaning for each of these descriptive terms.

1.1.1 Byte Stream Delivery

TCP interfaces between the application layer above and the network layer below. When an application sends data to TCP, it does so in 8-bit byte streams. It is then up to the sending TCP to segment or delineate the byte stream in order to transmit data in manageable pieces to the receiver[1]. It is this lack of 'record boundaries" which give it the name "byte stream delivery service".

1.1.2 Connection-Oriented

Before two communicating TCPs can exchange data, they must first agree upon the willingness to communicate. Analogous to a telephone call, a connection must first be made before two parties exchange information.

1.1.3 Reliability

A number of mechanisms help provide the reliability TCP guarantees. Each of these is described briefly below.

Checksums. All TCP segments carry a checksum, which is used by the receiver to detect errors with either the TCP header or data.

Duplicate data detection. It is possible for packets to be duplicated in packet switched network; therefore TCP keeps track of bytes received in order to discard duplicate copies of data that has already been received.[2]

Retransmissions. In order to guarantee delivery of data, TCP must implement retransmission schemes for data that may be lost or damaged. The use of positive acknowledgements by the receiver to the sender confirms successful reception of data. The lack of positive acknowledgements, coupled with a timeout period (see timers below) calls for a retransmission.

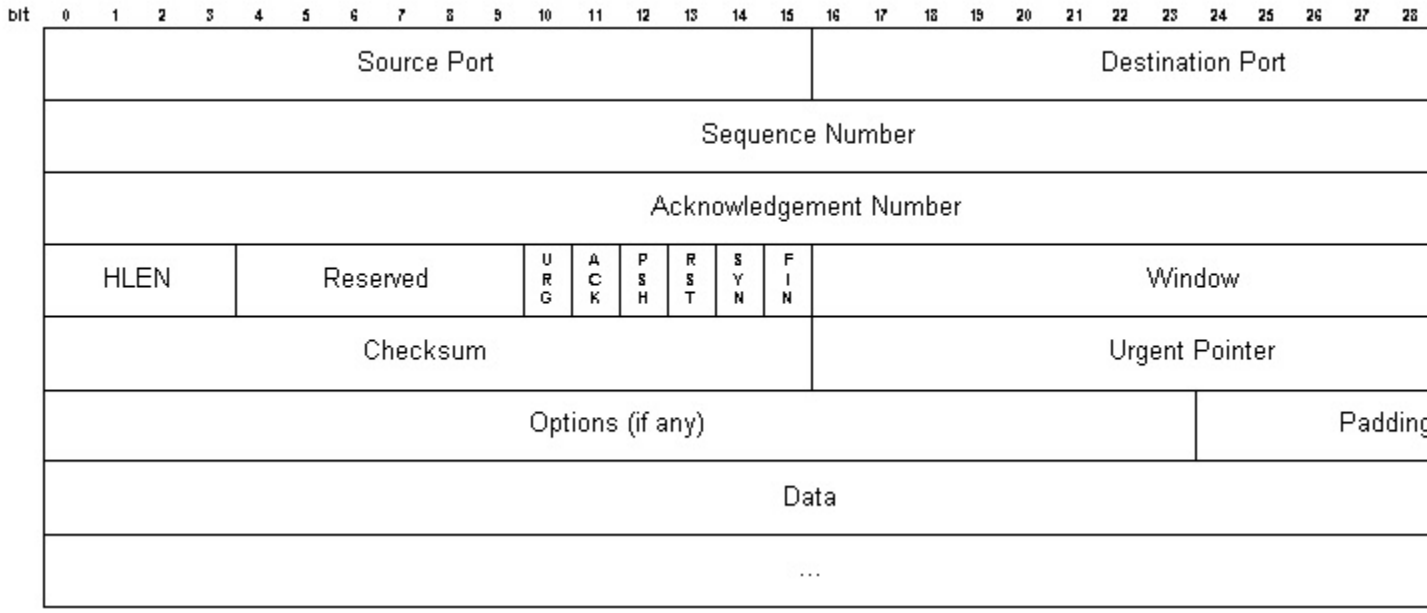
Sequencing. In packet switched networks, it is possible for packets to be delivered out of order. It is TCP's job to properly sequence segments it receives so it can deliver the byte stream data to an application in order.

Timers. TCP maintains various static and dynamic timers on data sent. The sending TCP waits for the receiver to reply with an acknowledgement within a bounded length of time. If the timer expires before receiving an acknowledgement, the sender can retransmit the segment.

1.2 TCP Header Format

Remember that the combination of TCP header and TCP in one packet is called a TCP segment. Figure 1 depicts the format of all valid TCP segments. The size of the header without options is 20 bytes. We will briefly define each field of the TCP header below.

Figure 1 - TCP Header Format



1.2.1 Source Port

A 16-bit number identifying the application the TCP segment originated from within the sending host. The port numbers are divided into three ranges, well-known ports (0 through 1023), registered ports (1024 through 49151) and private ports (49152 through 65535). Port assignments are used by TCP as an interface to the application layer. For example, the TELNET server is always assigned to the well-known port 23 by default on TCP hosts. A complete pair of IP addresses (source and destination) plus a complete pair of TCP ports (source and destination) define a single TCP connection that is globally unique. See [5] for further details.

1.2.2 Destination Port

A 16-bit number identifying the application the TCP segment is destined for on a receiving host. Destination ports use the same port number assignments as those set aside for source ports [5].

1.2.3 Sequence Number

A 32-bit number identifying the current position of the first data byte in the segment within the entire byte stream for the TCP connection. After reaching 2^{32} -1, this number will wrap around to 0.

1.2.4 Acknowledgement Number

A 32-bit number identifying the next data byte the sender expects from the receiver. Therefore, the number will be one greater than the most recently received data byte. This field is only used when the ACK control bit is turned on (see below).

1.2.5 Header Length

A 4-bit field that specifies the total TCP header length in 32-bit words (or in multiples of 4 bytes if you prefer). Without options, a TCP header is always 20 bytes in length. The largest a TCP header may be is 60 bytes. This field is required because the size of the options field(s) cannot be determined in advance. Note that this field is called "data offset" in the official TCP standard, but header length is more commonly used.

1.2.6 Reserved

A 6-bit field currently unused and reserved for future use.

1.2.7 Control Bits

Urgent Pointer (URG). If this bit field is set, the receiving TCP should interpret the urgent pointer field (see below).

Acknowledgement (ACK). If this bit field is set, the acknowledgement field described earlier is valid.

Push Function (PSH). If this bit field is set, the receiver should deliver this segment to the receiving application as soon as possible. An example of its use may be to send a Control-BREAK request to an application, which can jump ahead of queued data.

Reset the Connection (RST). If this bit is present, it signals the receiver that the sender is aborting the connection and all queued data and allocated buffers for the connection can be freely relinquished.

Synchronize (SYN). When present, this bit field signifies that sender is attempting to "synchronize" sequence numbers. This bit is used during the initial stages of connection establishment between a sender and receiver.

No More Data from Sender (FIN). If set, this bit field tells the receiver that the sender has reached the end of its byte stream for the current TCP connection.

1.2.8 Window

A 16-bit integer used by TCP for flow control in the form of a data transmission window size. This number tells the sender how much data the receiver is willing to accept. The maximum value for this field would limit the window size to 65,535 bytes, however a "window scale" option can be used to make use of even larger windows.

1.2.9 Checksum

A TCP sender computes a value based on the contents of the TCP header and data fields. This 16-bit value will be compared with the value the receiver generates using the same computation. If the values match, the receiver can be very confident that the segment arrived intact.

1.2.10 Urgent Pointer

In certain circumstances, it may be necessary for a TCP sender to notify the receiver of urgent data that should be processed by the receiving application as soon as possible. This 16-bit field tells the receiver when the last byte of urgent data in the segment ends.

1.2.11 Options

In order to provide additional functionality, several optional parameters may be used between a TCP sender and receiver. Depending on the option(s) used, the length of this field will vary in size, but it cannot be larger than 40 bytes due to the size of the header length field (4 bits). The most common option is the maximum segment size (MSS) option. A TCP receiver tells the TCP sender the maximum segment size it is willing to accept through the use of this option. Other options are often used for various flow control and congestion control techniques.

1.2.12 Padding

Because options may vary in size, it may be necessary to "pad" the TCP header with zeroes so that the segment ends on a 32-bit word boundary as defined by the standard [10].

1.2.13 Data

Although not used in some circumstances (e.g. acknowledgement segments with no data in the reverse direction), this variable length field carries the application data from TCP sender to receiver. This field coupled with the TCP header fields constitutes a TCP segment.

2. Connection Establishment and Termination

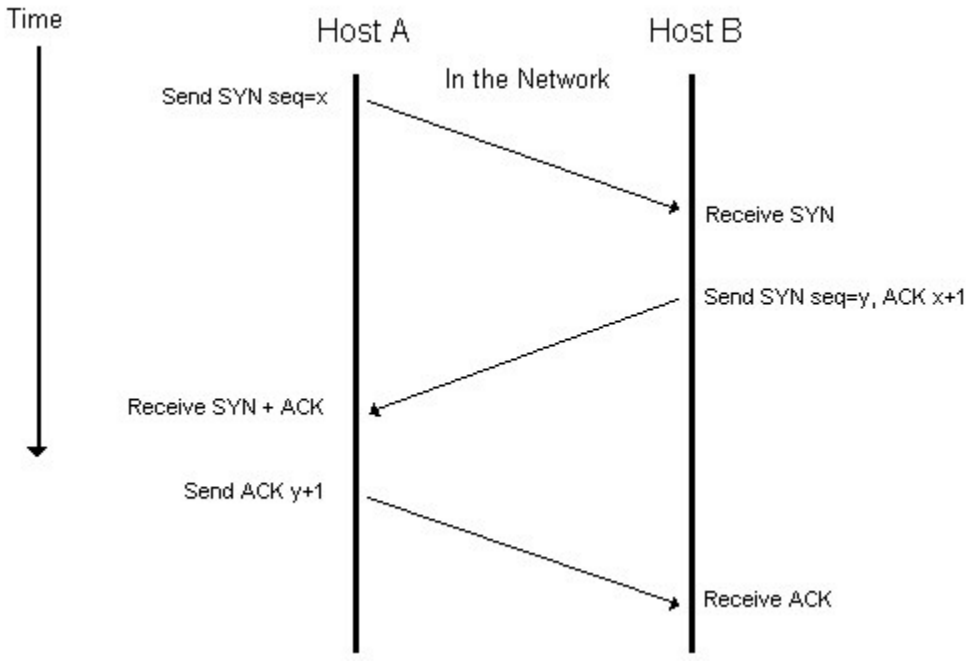
TCP provides a connection-oriented service over packet switched networks. Connection-oriented implies that there is a virtual connection between two endpoints.[3]

There are three phases in any virtual connection. These are the connection establishment, data transfer and connection termination phases.

2.1 Three-Way Handshake

In order for two hosts to communicate using TCP they must first establish a connection by exchanging messages in what is known as the three-way handshake. The diagram below depicts the process of the three-way handshake.

Figure 2 - TCP Connection Establishment



From figure 2, it can be seen that there are three TCP segments exchanged between two hosts, Host A and Host B. Reading down the diagram depicts events in time.

To start, Host A initiates the connection by sending a TCP segment with the SYN control bit set and an initial sequence number (ISN) we represent as the variable x in the sequence number field.

At some moment later in time, Host B receives this SYN segment, processes it and responds with a TCP segment of its own. The response from Host B contains the SYN control bit set and its own ISN represented as variable y. Host B also sets the ACK control bit to indicate the next expected byte from Host A should contain data starting with sequence number x+1.

When Host A receives Host B's ISN and ACK, it finishes the connection establishment phase by sending a final acknowledgement segment to Host B. In this case, Host A sets

the ACK control bit and indicates the next expected byte from Host B by placing acknowledgement number y+1 in the acknowledgement field.

In addition to the information shown in the diagram above, an exchange of source and destination ports to use for this connection are also included in each senders' segments.[4]

2.2 Data Transfer

Once ISNs have been exchanged, communicating applications can transmit data between each other. Most of the discussion surrounding data transfer requires us to look at flow control and congestion control techniques which we discuss later in this document and refer to other texts [9]. A few key ideas will be briefly made here, while leaving the technical details aside.

A simple TCP implementation will place segments into the network for a receiver as long as there is data to send and as long as the sender does not exceed the window advertised by the receiver. As the receiver accepts and processes TCP segments, it sends back positive acknowledgements, indicating where in the byte stream it is. These acknowledgements also contain the "window" which determines how many bytes the receiver is currently willing to accept. If data is duplicated or lost, a "hole" may exist in the byte stream. A receiver will continue to acknowledge the most current contiguous place in the byte stream it has accepted.

If there is no data to send, the sending TCP will simply sit idly by waiting for the application to put data into the byte stream or to receive data from the other end of the connection.

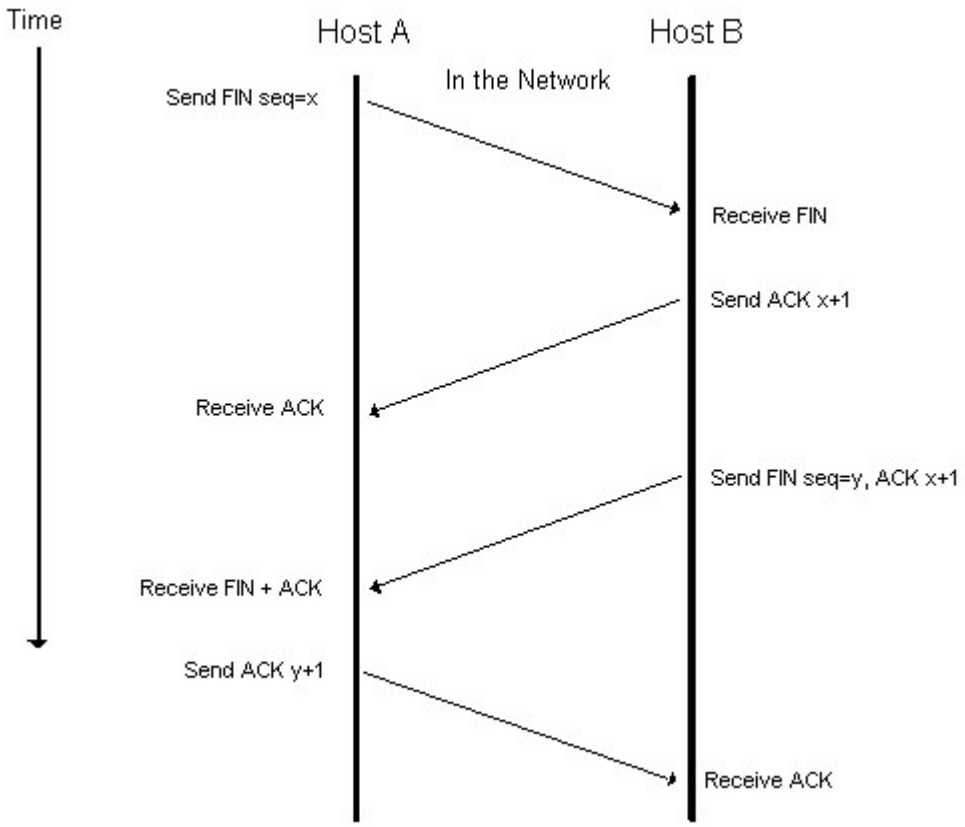
If data queued by the sender reaches a point where data sent will exceed the receiver's advertised window size, the sender must halt transmission and wait for further acknowledgements and an advertised window size that is greater than zero before resuming.

Timers are used to avoid deadlock and unresponsive connections. Delayed transmissions are used to make more efficient use of network bandwidth by sending larger "chunks" of data at once rather than in smaller individual pieces.[5]

2.3 Connection Termination

In order for a connection to be released, four segments are required to completely close a connection. Four segments are necessary due to the fact that TCP is a full-duplex protocol, meaning that each end must shut down independently.[6] The connection termination phase is shown in figure 3 below.

Figure 3 - TCP Connection Termination



Notice that instead of SYN control bit fields, the connection termination phase uses the FIN control bit fields to signal the close of a connection.

To terminate the connection in our example, the application running on Host A signals TCP to close the connection. This generates the first FIN segment from Host A to Host B. When Host B receives the initial FIN segment, it immediately acknowledges the segment and notifies its destination application of the termination request. Once the application on Host B also decides to shut down the connection, it then sends its own FIN segment, which Host A will process and respond with an acknowledgement.

3. Sliding Window and Flow Control

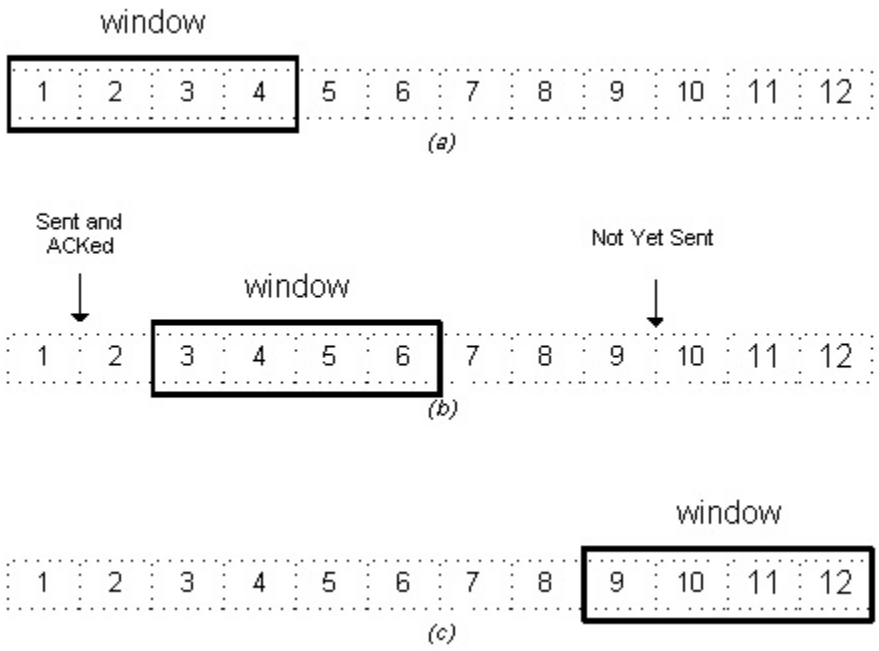
Flow control is a technique whose primary purpose is to properly match the transmission rate of sender to that of the receiver and the network. It is important for the transmission to be at a high enough rate to ensure good performance, but also to protect against overwhelming the network or receiving host.

In [8], we note that flow control is not the same as congestion control. Congestion control is primarily concerned with a sustained overload of network intermediate devices such as IP routers.

TCP uses the window field, briefly described previously, as the primary means for flow control. During the data transfer phase, the window field is used to adjust the rate of flow of the byte stream between communicating TCPs.

Figure 4 below illustrates the concept of the sliding window.

Figure 4 - Sliding Window



In this simple example, there is a 4-byte sliding window. Moving from left to right, the window "slides" as bytes in the stream are sent and acknowledged.[7] The size of the window and how fast to increase or decrease the window size is an area of great research. We again refer to other documents for further detail [9].

Introduction:

The OSI model consists of seven protocol layers and each layer performs a supportive communication task.
The Transport layer is the fourth layer in the OSI model, which provides communication services between the computers connected in the network.
For example: The transport layer provides an error checking service during the transmission of data packets from source computer to destination computer.

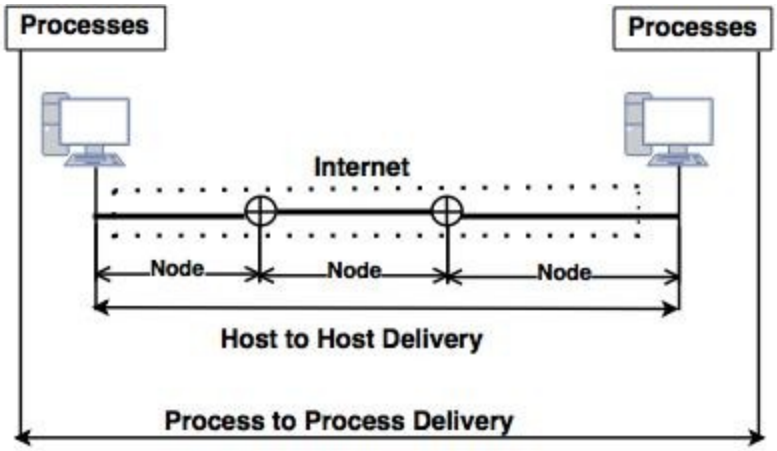
Process-to-Process Delivery

- The data link layer helps to deliver the frames between two neighboring nodes over a link. This process is called as node-to-node delivery.

- The network layer helps to deliver the datagrams between two hosts. This process is called as host-to-host delivery.

- Several processes are carried out on the source host and destination host. Some mechanism is needed to complete the delivery process between the source host and destination host.

- The transport layer helps to carry out the process-to-process delivery i.e. the delivery of a packet or part of message from one process to another process.



Process to Process Delivery