TCP(Transmission Control Protocol)

Like UDP, is a process-to-process (program-to-program) protocol. TCP, therefore, like UDP, uses port numbers. Unlike UDP, TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

TCP Services

Process-to-Process Communication: Like UDP, TCP provides process-to-process communication using port numbers.

Well-known ports used by TCP

Port	***Protocol***	***Description***
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Stream Delivery Service: TCP, unlike UDP, is a stream-oriented protocol. TCP allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.

Sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction.

Full-Duplex Communication: TCP offers full-duplex service, in which data can flow in both directions at the same time.

Connection-Oriented Service: TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two TCPs establish a connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.

Note that this is a virtual connection, not a physical connection. The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost, or corrupted, and then resent. Each may use a different path to reach the destination. There is no physical connection.

Reliable Service: TCP is a reliable transport protocol. It uses an **acknowledgment** mechanism to check the safe and sound arrival of data.

TCP Features

- *Numbering System:* Although the TCP software keeps track of the segments being transmitted or received, there is no field for a segment number value in the segment header. Instead, there are two fields called the sequence number and the acknowledgment number. These two fields refer to the byte number and not the segment number.
  - *Byte Number* TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it

stores them in the sending buffer and numbers them. The numbering does not necessarily start from 0. Instead, TCP generates a random number between 0 and $2^{32} - 1$ for the number of the first byte. For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056. We will see that byte numbering is used for flow and error control.

- *Sequence Number* After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment

  Example : *Segment 1 Sequence Number: 10,001 (range: 10,001 to 11,(00)*
  Segment 2 Sequence Number: 11,001 (range: 11,001 to 12,000)
  Segment 3 Sequence Number: 12,001 (range: 12,001 to 13,000)
  Segment 4 Sequence Number: 13,001 (range: 13,001 to 14,000)

- *Acknowledgment Number:* The value of the acknowledgment field in a segment defines the number of the **next byte** a party expects to receive. The acknowledgment number is cumulative.

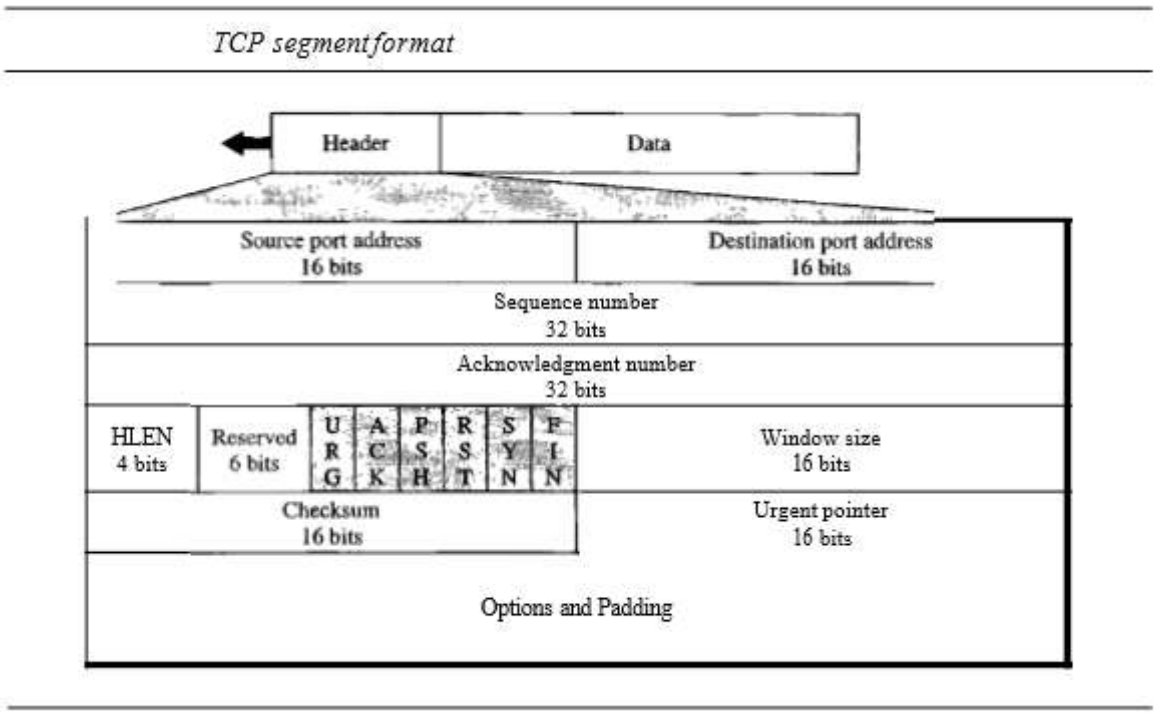
- *Flow Control:* TCP, unlike UDP, provides *flow control.* The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

- *Error Control:* To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

- *Congestion Control:* TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network.

Segment

A packet in TCP is called a **segment.**



Format: The segment consists of a 20 to 60 byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options. Some of the header fields are

- *Source port address*. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment. This serves the same purpose as the source port address in the UDP header.

- *Destination port address*. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment. This serves the same purpose as the destination port address in the UDP header.
- *Sequence number*. This 32-bit field defines the number assigned to the first byte of data contained in this segment.
- *Acknowledgment number*. This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number *x* from the other party, it defines *x* + 1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- *Header length*. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes*Reserved*. This is a 6-bit field reserved for future use.
- *Control*. This field defines 6 different control bits or flags as shown below. One or more of these bits can be set at a time.
- *Window size*. This field defines the size of the window, in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes.

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

- *Checksum.* This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory
- *Urgent pointer*. This l6-bit field, which is valid, only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
- *Options.* There can be up to 40 bytes of optional information in the TCP header.

A TCP Connection

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All the segments belonging to a message are then sent over this virtual path.

In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

1. Connection Establishment

TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously. This implies that each party must initialize communication and get approval from the other party before any data are transferred.

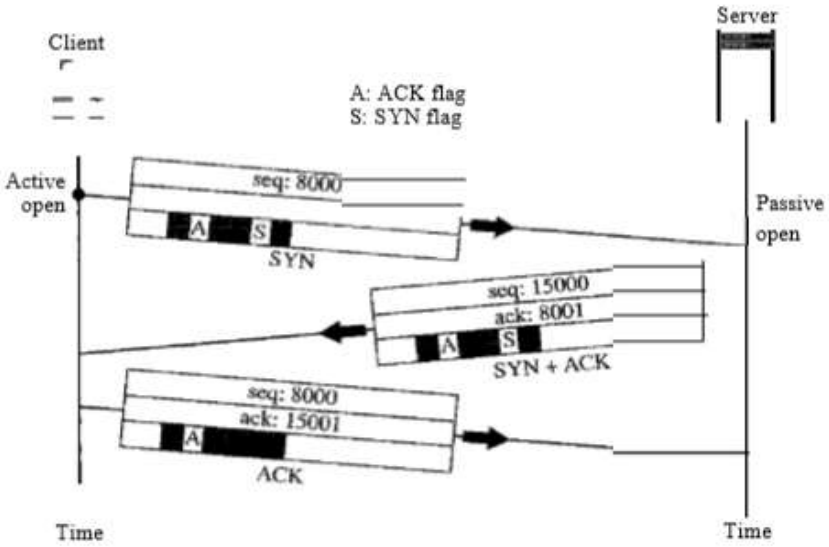
Three-Way Handshaking

The 3-Way Handshake process is the defined set of steps that takes place in the TCP for creating a secure and reliable communication link and also closing it. Actually, TCP uses the 3-way handshake process to establish a connection between two devices before transmitting the data. After the establishment of the connection, the data transfer takes place between the devices. After which the connection needs to be terminated, which is also done by using the 3-way handshake process. The secure and reliable connection is established to reserve the CPU, buffer, and bandwidth of the devices to communicate properly. Thus, it is a must to free these resources by terminating the connection after data transmission. Hence, the TCP 3-way handshake process can be used to establish and terminate connections in the network in a secure way.

Threeway handshaking process: In our example, an application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport layer protocol. The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a **passive open**. Although the server

TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself. The client program issues a request for an **active open**. A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as .

Connection establishment using three-way handshaking



The three steps in this phase are as follows.

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers

2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK. This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment. It consumes one sequence number.

3. The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. Note that the sequence number in this segment is the same as the one in the SYN segment

Simultaneous Open A rare situation, called a simultaneous open, may occur when both processes issue an active open. In this case, both TCPs transmit a SYN + ACK segment to each other, and one single connection is established between them.

SYN Flooding Attack The connection establishment procedure in TCP is susceptible to a serious security problem called the ***SYN flooding attack***. This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is corning from a different client. The server, assuming that the clients are issuing an active open. The TCP server then sends the SYN +ACK segments to the fake clients, which are lost. During this time, however, a lot of resources are occupied without being used. If, during this short time, the number of SYN segments is large, the server eventually runs out of resources and may crash. This SYN flooding attack belongs to a type of security attack known as a **denial-of-service attack**, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request. Some implementations of TCP have strategies to alleviate the effects of a SYN attack

2.Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments.

Pushing Data We saw that the sending TCP uses a buffer to store the stream of data coming from the sending application program.. The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP . consider an application program where client site wants to send a keystroke to the application at the other site and receive an immediate response. TCP can

handle such a situation. The application program at the sending site can request a **push operation**. This means that the sending TCP must not wait for the buffer to be filled. It must create a segment and send it immediately. The sending TCP must also set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come. TCP can choose whether or not to use this feature.

Urgent Data TCP is a stream-oriented protocol. This means that the data are presented from the application program to TCP as a stream of bytes. Each byte of data has a position in the stream. However, on occasion an application program needs to send urgent bytes.

. The solution is to send a segment with the URG bit set. The sending application program tells the sending TCP that the piece of data is urgent. The sending TCP creates a segment and inserts the urgent data at the beginning of the segment. The rest of the segment can contain normal data from the buffer. When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment..

3.Connection Termination

Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client

Most implementations today allow three-way handshaking for connection termination as

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a **FIN segment** in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure 23.20. If it is only a control segment, it consumes only one sequence number.

2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN +ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction.

This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server

Half-Close

In TCP, one end can stop sending data while still receiving data. This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client. The client half-closes the connection by sending a FIN segment. The server accepts the half-close by sending the ACK segment. The data transfer from the client to the server stops. The server, however, can still send data. When the server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client. After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server. The client cannot send any more data to the server.