Advanced Computer Network (4350706)

Unit-4
Transport Layer Protocols

Prepared By :
Mr. Chetan C. Kamani
Lecturer, Computer Engineering Department
Government Polytechnic, Jamnagar

Topics to be discussed

- Transmission Control Protocol
  - TCP Services
  - TCP Features
  - Segment
  - A TCP Connection

Transmission Control Protocol

- Transmission Control Protocol (TCP) is a ***connection-oriented, reliable protocol.***
- TCP explicitly defines ***connection establishment, data transfer, and connection teardown phases*** to provide a connection-oriented service.
- TCP uses a combination of ***GBN and SR protocols to provide reliability***. To achieve this goal, TCP uses checksum (for error detection), retransmission of lost or corrupted packets, cumulative and selective acknowledgments, and timers.
- In this section, we first discuss the services provided by TCP; we then discuss the TCP features in more detail. TCP is the most common transport-layer protocol in the Internet.

TCP Services

- Process-to-Process Communication
- Stream Delivery Service
  - Sending and Receiving Buffers
  - Segments
- Full-Duplex Communication
- Multiplexing and Demultiplexing
- Connection-Oriented Service
- Reliable Service

Process-to-Process Communication

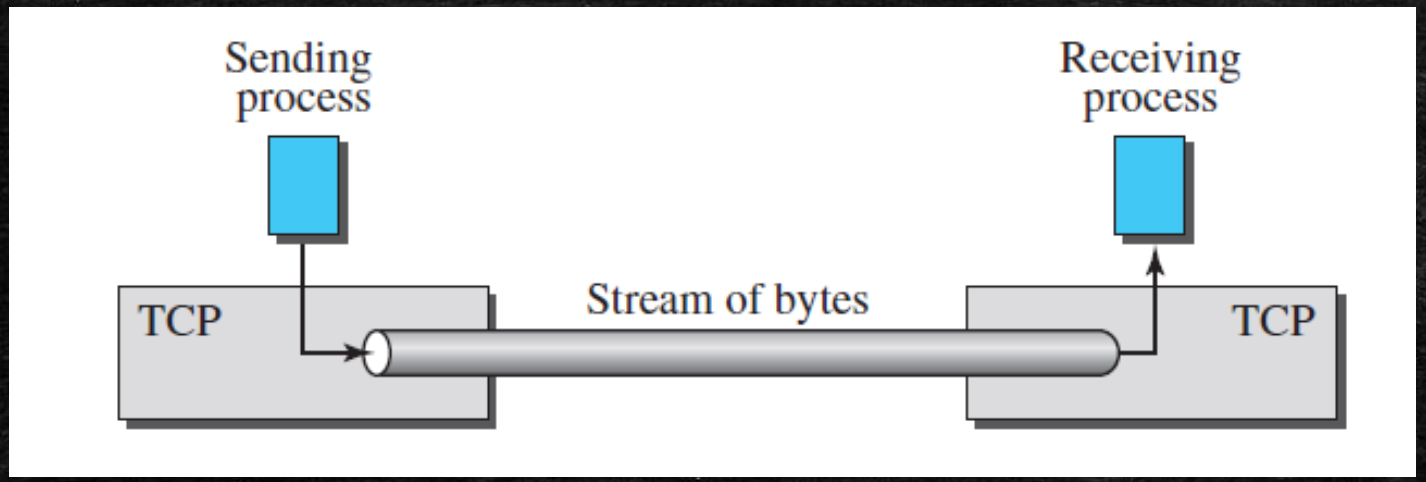
- As with UDP, TCP provides process-to-process communication using port numbers.

Stream Delivery Service

- TCP, unlike UDP, is a stream-oriented protocol.
- In UDP, a process sends messages with predefined boundaries to UDP for delivery.
- UDP adds its own header to each of these messages and delivers it to IP for transmission.
- Each message from the process is called a user datagram, and becomes, eventually, one IP datagram.
- Neither IP nor UDP recognizes any relationship between the datagrams.

Stream Delivery Service

- TCP, on the other hand, 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.
- TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their bytes across the Internet.
- This imaginary environment is depicted in Figure. The sending process produces (writes to) the stream and the receiving process consumes (reads from) it.

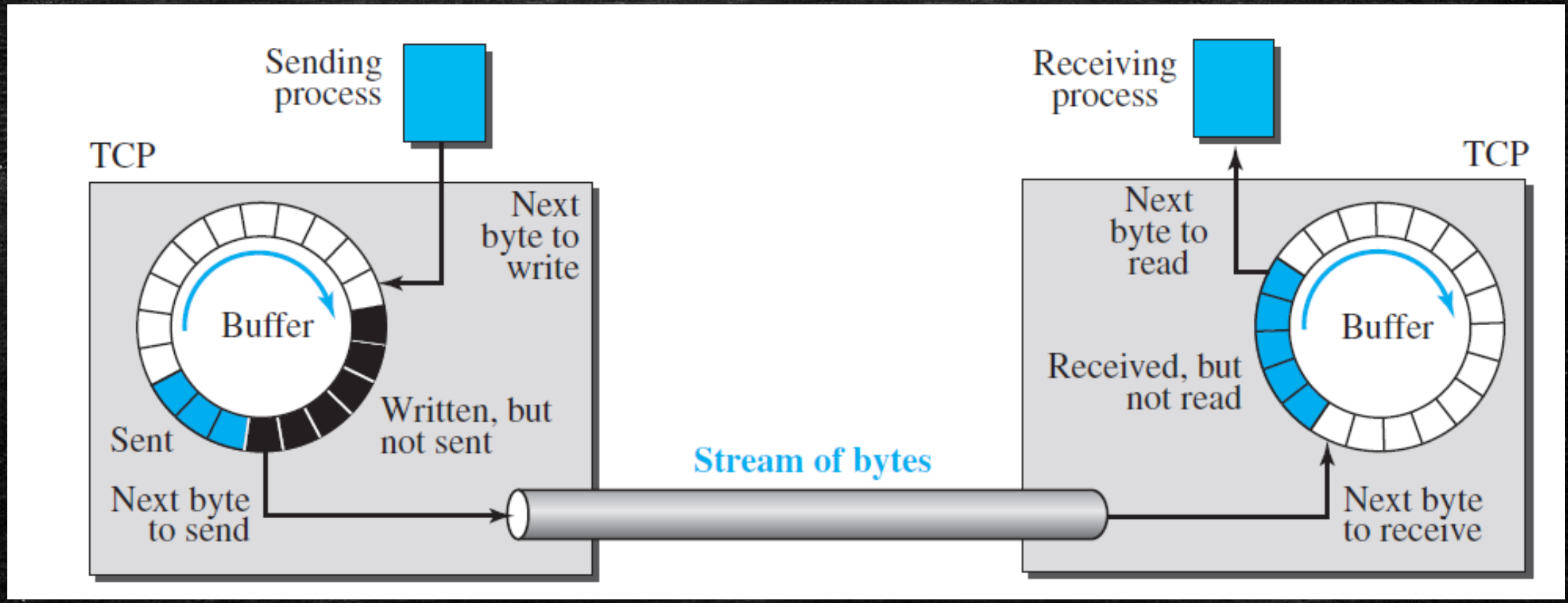


Stream delivery

Stream Delivery Service

- Because the sending and the receiving processes may not necessarily write or read data at the same rate, TCP needs buffers for storage.
- There are two buffers, the sending buffer and the receiving buffer, one for each direction.
- These buffers are also necessary for flow- and error-control mechanisms used by TCP.
- One way to implement a buffer is to use a circular array of 1-byte locations as shown in Figure.
- For simplicity, we have shown two buffers of 20 bytes each; normally the buffers are hundreds or thousands of bytes, depending on the implementation.
- We also show the buffers as the same size, which is not always the case.

Stream Delivery Service

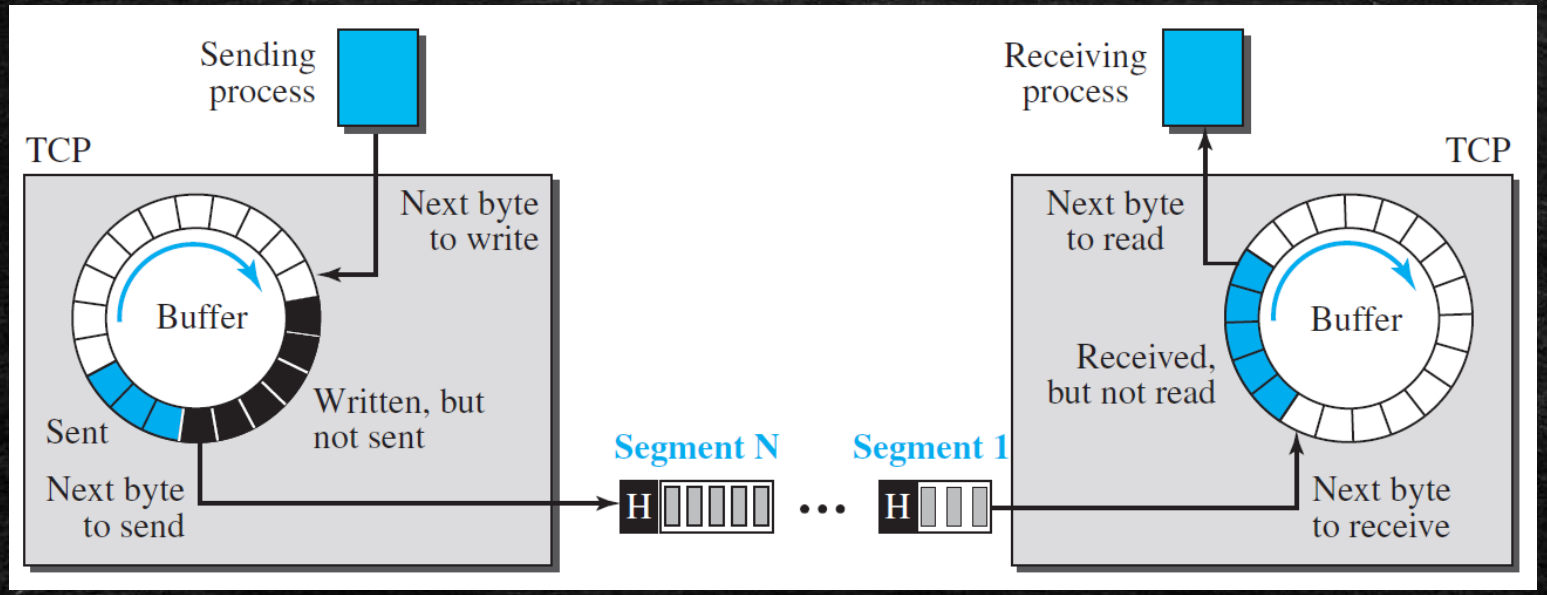


Stream Delivery Service

- Although buffering handles the disparity between the speed of the producing and consuming processes, we need one more step before we can send data.
- The network layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes.
- At the transport layer, TCP groups a number of bytes together into a packet called a segment. TCP adds a header to each segment (for control purposes) and delivers the segment to the network layer for transmission.
- The segments are encapsulated in an IP datagram and transmitted.
- This entire operation is transparent to the receiving process.

Stream Delivery Service

- Following Figure shows how segments are created from the bytes in the buffers.
- Note that segments are not necessarily all the same size. In the figure, for simplicity, we show one segment carrying 3 bytes and the other carrying 5 bytes.
- In reality, segments carry hundreds, if not thousands, of bytes.



TCP segments

Full-Duplex Communication

- TCP offers full-duplex service, where data can flow in both directions at the same time.
- Each TCP endpoint then has its own sending and receiving buffer, and segments move in both directions

Multiplexing and Demultiplexing

- Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver.
- However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes.

Connection-Oriented Service

- TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:
  1. The two TCP's establish a logical connection between them.
  2. Data are exchanged in both directions.
  3. The connection is terminated.
- Note that this is a logical 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 be routed over a different path to reach the destination. There is no physical connection.
- TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.

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
  - Byte Number
  - Sequence Number
  - Acknowledgment Number

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 a byte number and not a segment number.

Byte Number (Numbering System)

- TCP numbers all data bytes (octets) that are transmitted in a connection.
- Numbering is independent in each direction.
- When TCP receives bytes of data from a process, TCP stores them in the sending buffer and numbers them. The numbering does not necessarily start from 0. Instead, TCP chooses an arbitrary number between 0 and $2^{32} - 1$ for the number of the first byte.
- For example, if the number happens to be 1057 and the total data to be sent is 6000 bytes, the bytes are numbered from 1057 to 7056.

Sequence Number (Numbering System)

- After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.
- The sequence number, in each direction, is defined as follows:
  1. The sequence number of the first segment is the ISN (initial sequence number), which is a random number.
  2. The sequence number of any other segment is the sequence number of the previous segment plus the number of bytes (real or imaginary) carried by the previous segment.
- Later, we show that some control segments are thought of as carrying one imaginary byte.

Sequence Number (Numbering System)

- Example
  - ***Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 10001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?***

Sequence Number (Numbering System)

Solution

The following shows the sequence number for each segment:

Segment 1	→	Sequence Number:	10001	**Range:**	10001	to	11000
Segment 2	→	Sequence Number:	11001	**Range:**	11001	to	12000
Segment 3	→	Sequence Number:	12001	**Range:**	12001	to	13000
Segment 4	→	Sequence Number:	13001	**Range:**	13001	to	14000
Segment 5	→	Sequence Number:	14001	**Range:**	14001	to	15000

Sequence Number (Numbering System)

- When a segment carries a combination of data and control information (piggybacking), it uses a sequence number.
- If a segment does not carry user data, it does not logically define a sequence number. The field is there, but the value is not valid.
- However, some segments, when carrying only control information, need a sequence number to allow an acknowledgment from the receiver. These segments are used for connection establishment, termination, or abortion.
- Each of these segments consume one sequence number as though it carries one byte, but there are no actual data. We will elaborate on this issue when we discuss connections.

Acknowledgment Number (Numbering System)

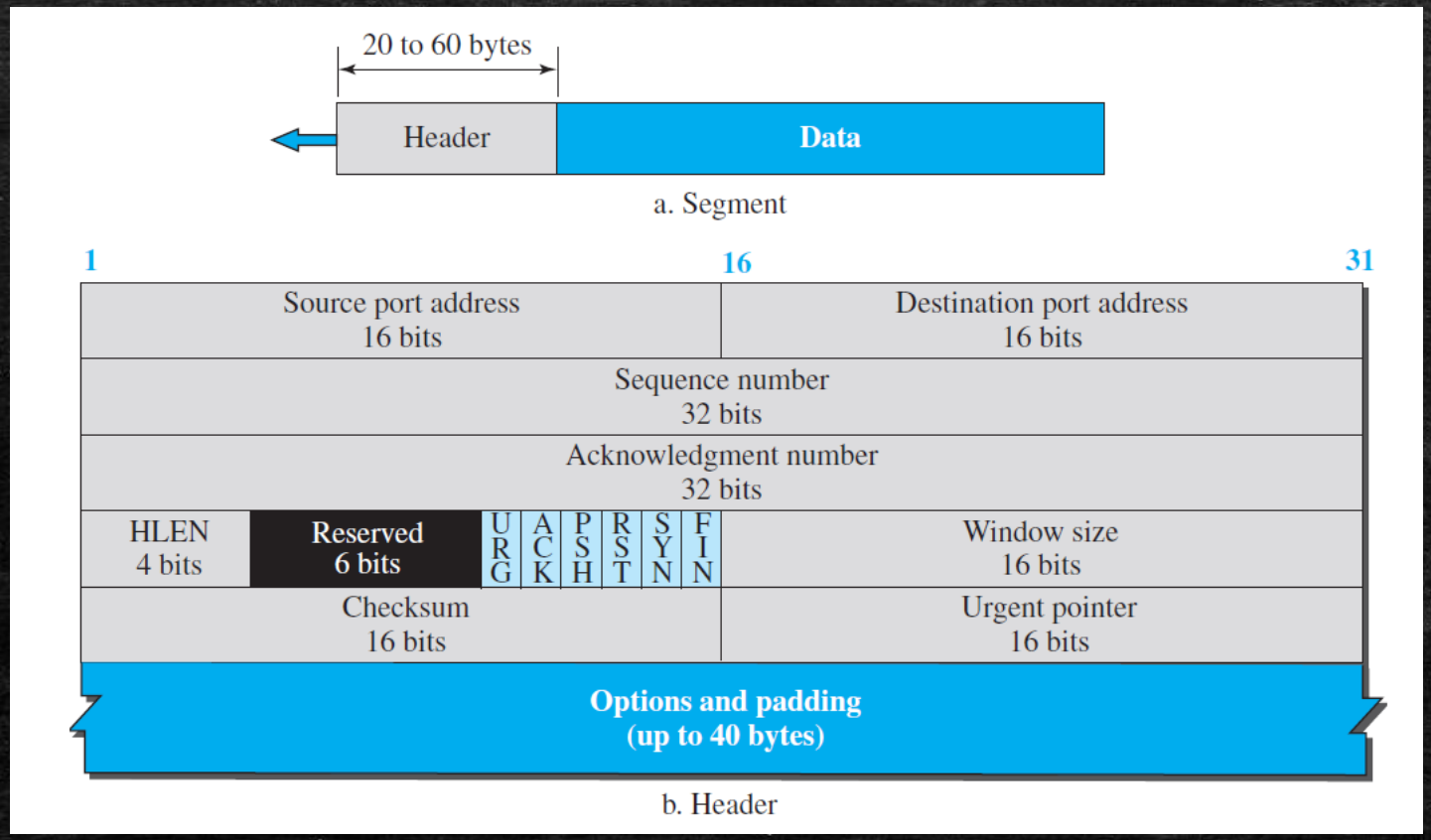
- As we discussed previously, communication in TCP is full duplex; when a connection is established, both parties can send and receive data at the same time.
- Each party numbers the bytes, usually with a different starting byte number.
- The sequence number in each direction shows the number of the first byte carried by the segment.

Acknowledgment Number (Numbering System)

- Each party also uses an acknowledgment number to confirm the bytes it has received.
- However, the acknowledgment number defines the number of the next byte that the party expects to receive.
- In addition, the acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number.
- The term cumulative here means that if a party uses 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642. Note that this does not mean that the party has received 5642 bytes, because the first byte number does not have to be 0.

Segment

- A packet in TCP is called a segment.



TCP segment format

Segment

- **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.
- **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.

Segment

- **Sequence number.**
  - This 32-bit field defines the number assigned to the first byte of data contained in this segment.
  - As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment. During connection establishment (discussed later) each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

Segment

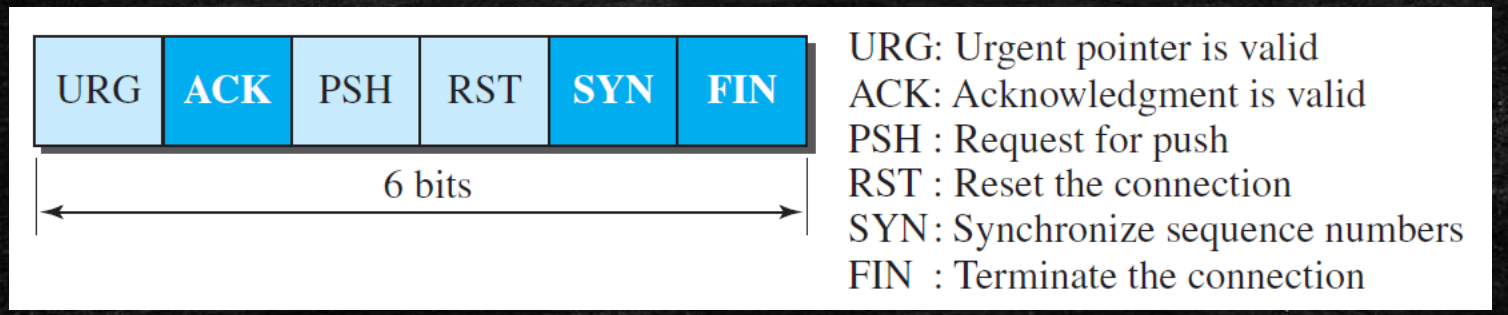
- **Acknowledgment number.**
  - This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party.
  - If the receiver of the segment has successfully received byte number x from the other party, it returns x + 1 as the acknowledgment number.
  - Acknowledgment and data can be piggybacked together.

Segment

- **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. Therefore, the value of this field is always between 5 (5 × 4 = 20) and 15 (15 × 4 = 60).

Segment

- **Control.**
  - This field defines 6 different control bits or flags, as shown in Figure 24.8.
  - One or more of these bits can be set at a time.
  - These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.
  - A brief description of each bit is shown in the figure.
  - We will discuss them further when we study the detailed operation of TCP later in the chapter.



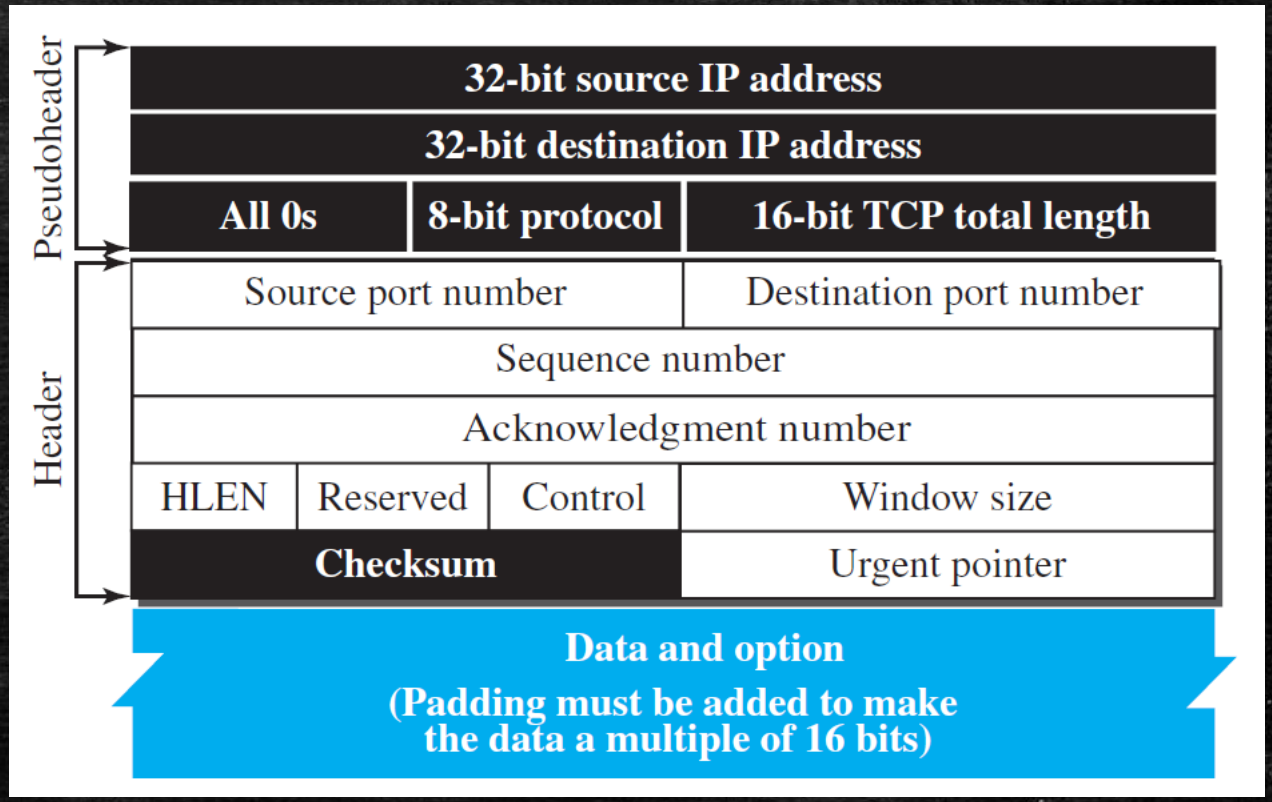
Control field

Segment

- **Window size.**
  - This field defines the window size of the sending TCP 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. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

Segment

- **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 use of the checksum in the UDP datagram is optional, whereas the use of the checksum for TCP is mandatory.
  - The same pseudoheader, serving the same purpose, is added to the segment.
  - For the TCP pseudoheader, the value for the protocol field is 6. See Figure.



Segment

- **Urgent pointer.**
  - This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
  - It defines a value 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.

Encapsulation

- *A TCP segment encapsulates the data received from the application layer.*
- *The TCP segment is encapsulated in an IP datagram, which in turn is encapsulated in a frame at the data-link layer.*

A TCP Connection

- Connection Establishment
  - Three-way Handshaking
  - SYN Flooding Attack
- Data Transfer
  - Pushing Data
  - Urgent Data
- Connection Termination
  - Three-way Handshaking
- Connection Reset

A TCP Connection

- TCP is connection-oriented. As discussed before, a connection-oriented transport protocol establishes a logical path between the source and destination.
- All of the segments belonging to a message are then sent over this logical path. Using a single logical pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames. You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented. The point is that a TCP connection is logical, not physical.
- TCP operates at a higher level. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost or corrupted, it is retransmitted.

A TCP Connection

- TCP operates at a higher level.
- TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself.
- If a segment is lost or corrupted, it is retransmitted. Unlike TCP, IP is unaware of this retransmission. If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering.
- In TCP, connection-oriented transmission requires three phases:
  - ***connection establishment,***
  - ***data transfer,***
  - ***and connection termination.***

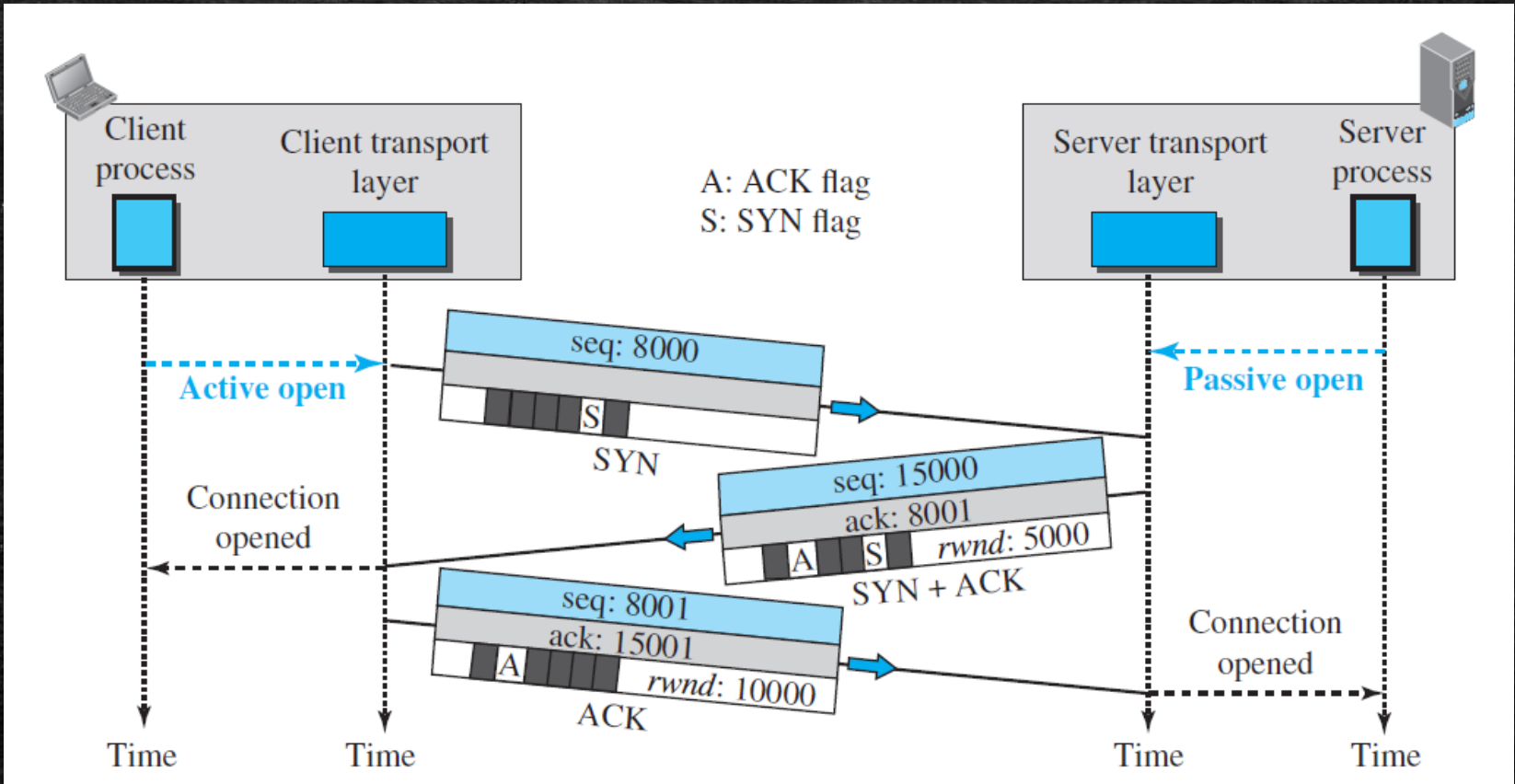
Connection Establishment

- 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.

Connection Establishment

- The connection establishment in TCP is called three-way handshaking.
- 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 request is called a passive open. Although the server TCP is ready to accept a 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 to connect to a particular server. TCP can now start the three-way handshaking process, as shown in Figure.

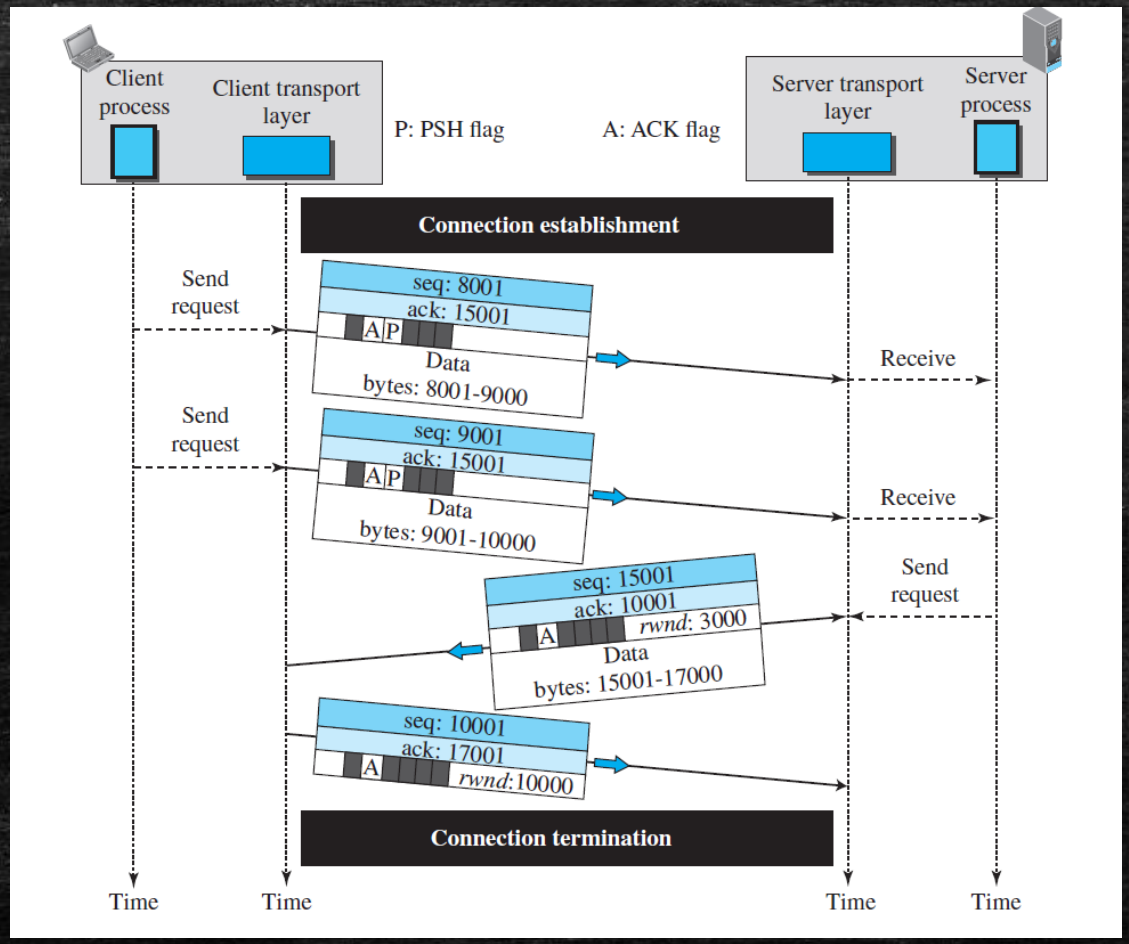
Connection Establishment



State Transition Diagram

- SYN Flooding Attack

Data Transfer



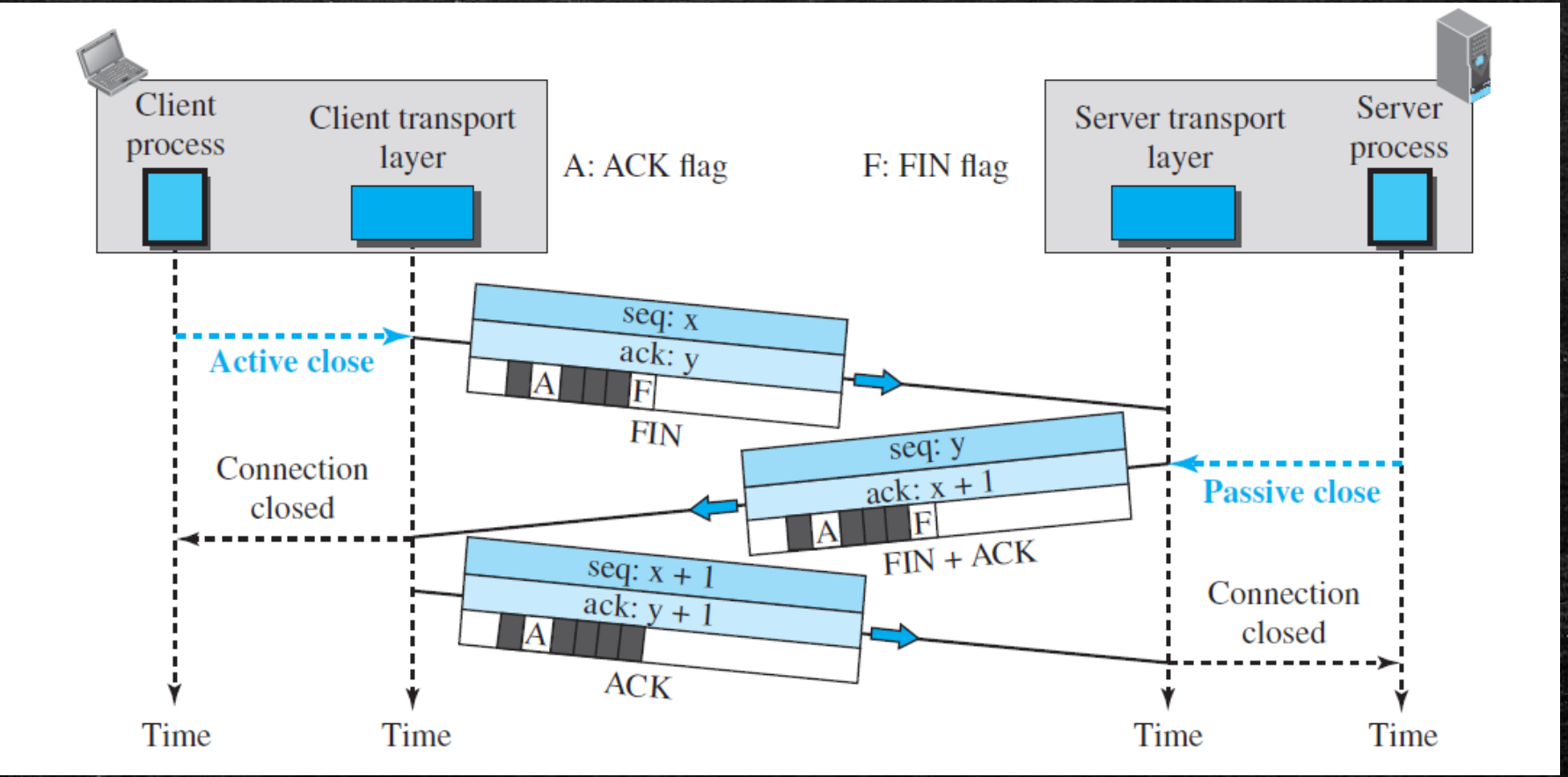
Data Transfer

- Pushing Flag
- Urgent Data

Connection Termination

- Either 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 two options for connection termination:
  - three-way handshaking
  - and four-way handshaking with a half-close option.

Connection Termination

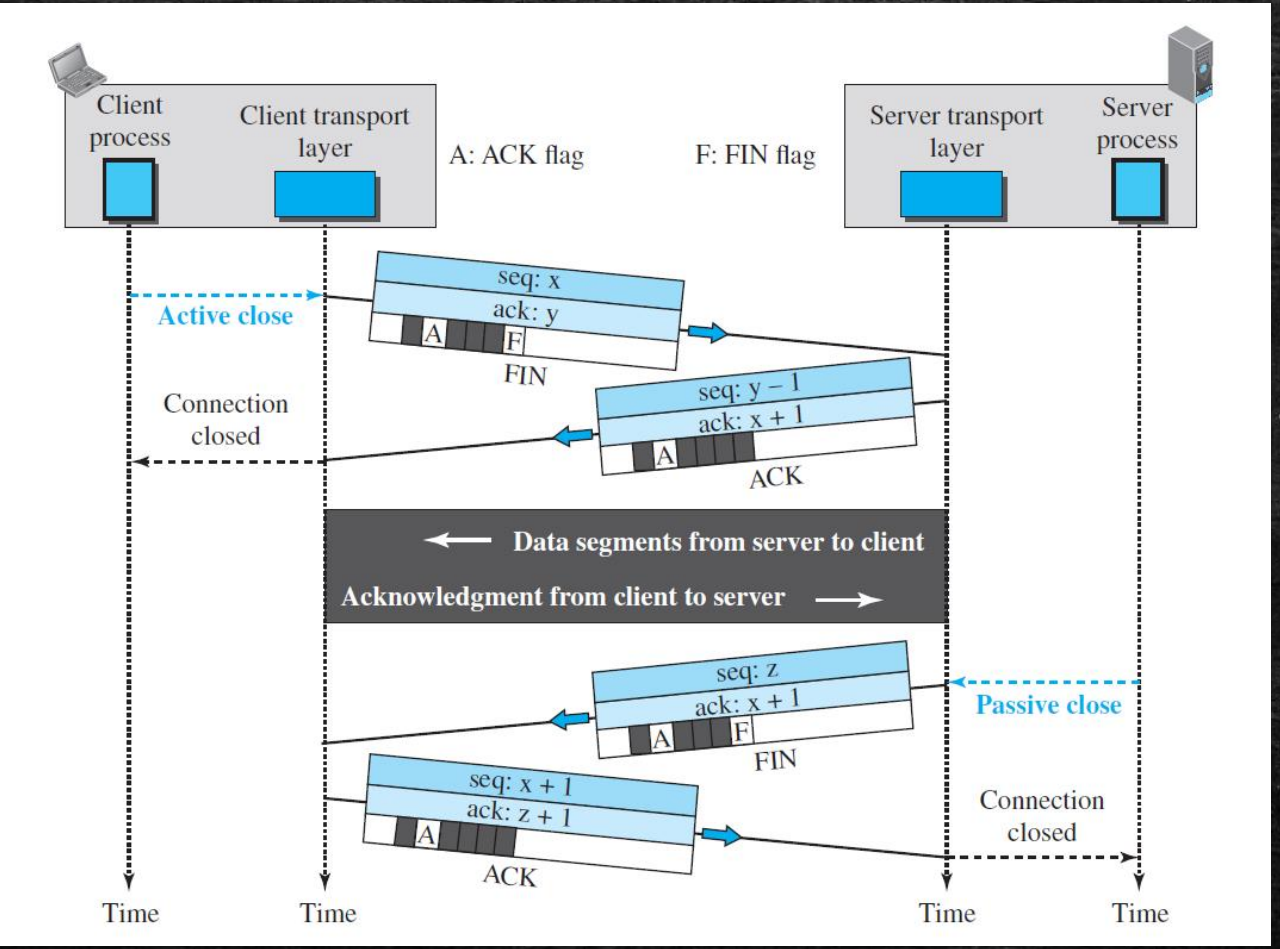


Connection termination using three-way handshaking

Connection Termination

- Half Close
  - In TCP, one end can stop sending data while still receiving data. This is called a halfclose.
  - Either the server or the client can issue a half-close request. It can occur when the server needs all the data before processing can begin. A good example is sorting.
  - When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start. This means the client, after sending all data, can close the connection in the client-to-server direction.
  - However, the server-to-client direction must remain open to return the sorted data. The server, after receiving the data, still needs time for sorting; its outbound direction must remain open.

Connection Termination



References

- Data Communication and Networking, Fifth Edition, BEHROUZ A. FOROUZAN
- Data Communication and Networking, Fourth Edition, BEHROUZAN A. FOROUZA
- https://www.geeksforgeeks.org/introduction-of-classful-ip-addressing/

Thank You