Session 3A

TCP: Connection Establishment

Mouli Sankaran

Session 3A: Focus

- TCP Connection Management
  - Control Bits: SYN and FIN
  - Quiz 1
- TCP Connection Establishment
  - Three-way handshake
  - Connection Set-up
  - Sequence and Acknowledgement Numbers
- TCP State Transition Diagram
  - Active and Passive Open (client and Server)
  - Simultaneous Open from both machines

Course page where the course materials will be posted as the course progresses:

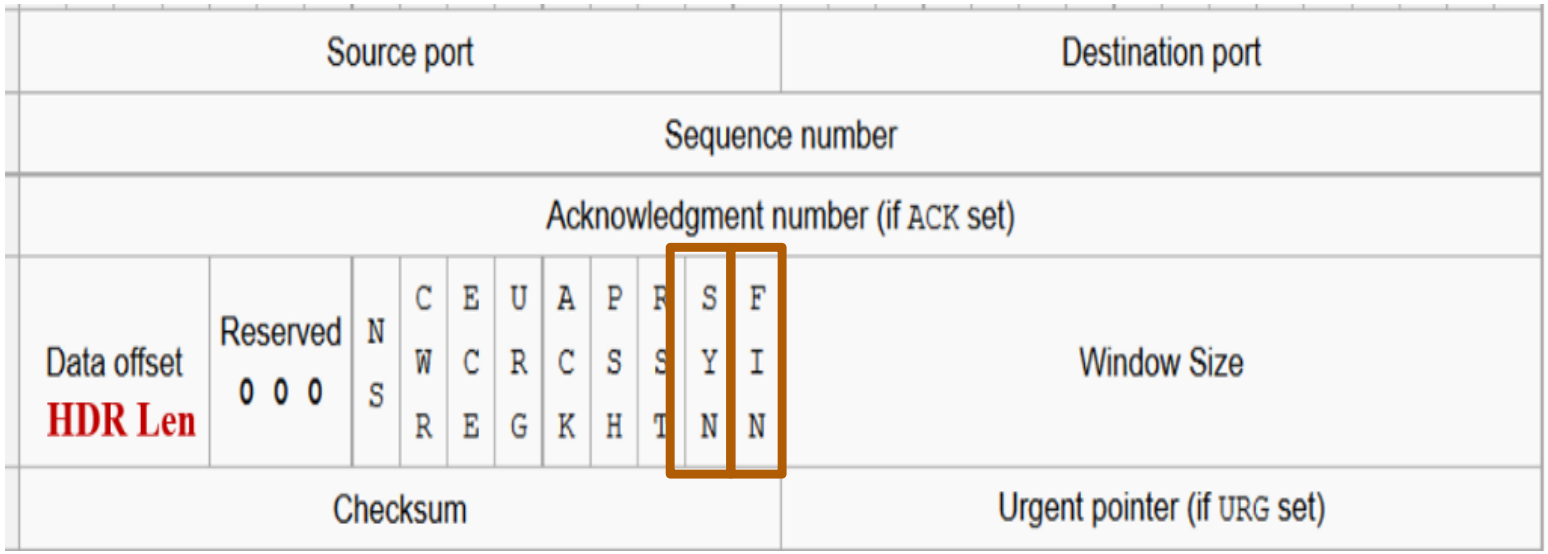
TCP Connection Establishment

Reference: Ref1: TCP/IP Illustrated-Volume 1:
Chapter 13: TCP Connection Establishment and Termination

TCP: Connection Management

- Let us now take a detailed look at what a TCP connection is, how it is established, and how it is terminated.
- Recall that TCP's service model is a **byte stream**.
- TCP detects and repairs essentially all the data transfer problems that may be introduced by packet loss, duplication, or errors at the IP layer below.
- Because of its management of ***connection state*** (information about the connection kept by both endpoints), TCP is a considerably more complicated protocol than UDP.
  - UDP is a *connectionless* protocol that involves no connection establishment or termination.
- In TCP, during connection establishment, several *options* can be exchanged between the two endpoints regarding the parameters of the connection. (Example: Window size, Maximum Segment size, etc.)

Connection Establishment: Control Bits



- When a **new connection** is being **established**, the **SYN bit** field is **turned on** in the **first segment** sent from **client** to **server**.
  - Such segments are called **SYN segments**, or simply **SYNs.**
- The **FIN flag** indicates the **end of data transmission** to **finish** a **TCP connection**.
  - The sender of the segment is finished sending data to its peer

SYN: It is a **short-form** of **Synch** or **Synchronization**

FIN: It is a **short-form** of the **Final segment** of a **TCP connection.**

Quiz 1: SYN and FIN Flags

What should the TCP/IP stack do when it receives a TCP segment with both SYN and FIN flags set from the network?

A. Consider it as a new connection request and termination of the last existing connection.

B. Consider only the SYN flag as valid and ignore FIN flag.

C. Consider only the FIN flag as valid and ignore SYN flag.

D. Drop the segment and take no action, since having both SYN and FIN flags set is an invalid condition.

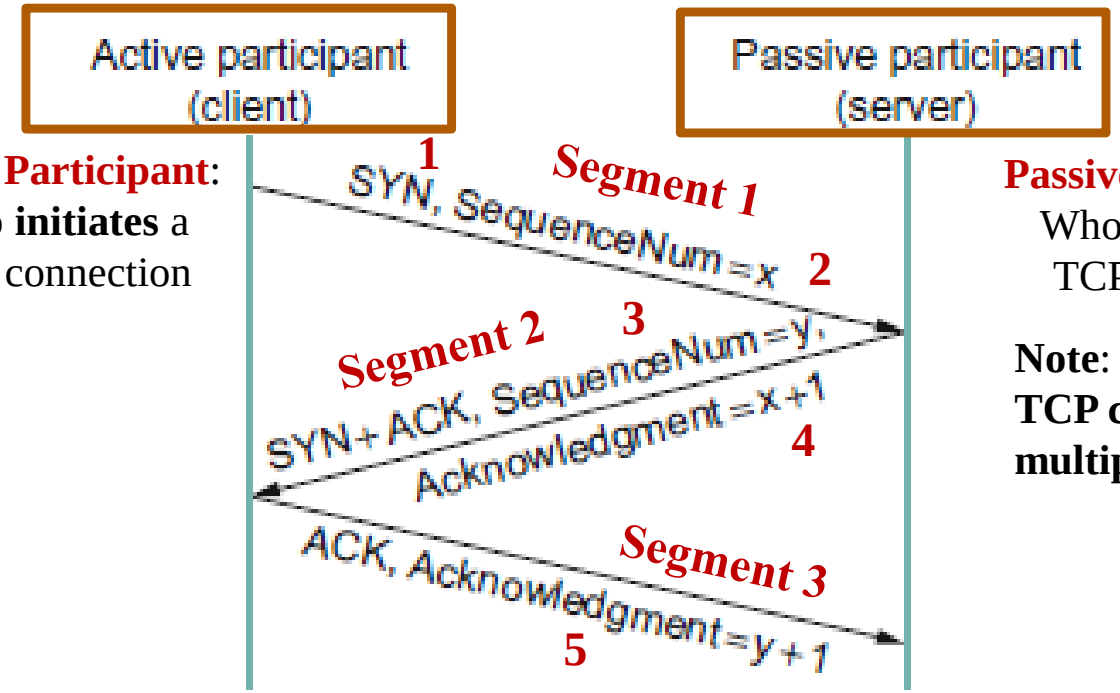
ANS: D

Note: Their **purposes** are **mutually exclusive**. A TCP header with **both** the **SYN** and **FIN flags set** is **anomalous TCP behavior**, causing various responses from the recipient, depending on the particular implementation of TCP/IP stack in an OS.

Interesting note :An attacker can send a segment with both flags set to see what kind of system reply is returned and thereby determine what kind of OS is on the receiving end. The attacker can then use any known system vulnerabilities for further attacks For example. **JunOS** checks if the SYN and FIN flags are set in TCP headers. If it discovers such a header, it drops the packet.

JunOS: It is an **OS** from **Juniper Networks**, running on Network equipment from Juniper.

TCP Connection Set-Up: Timeline



Active Participant: Who **initiates** a TCP connection

Passive Participant: Who **accepts** the TCP connection

Note: Normally **Servers** wait for **TCP connections** to be **initiated** by **multiple clients**.

1. SYN packet from **Client** with **SYN bit set**.

2. Seq no.(x): **Initial Sequential Number** from the **Client**

3. Seq no. (y): **Initial Sequential Number** from the **Server**

4a. ACK Acknowledgement from the Server with ACK num = x + 1

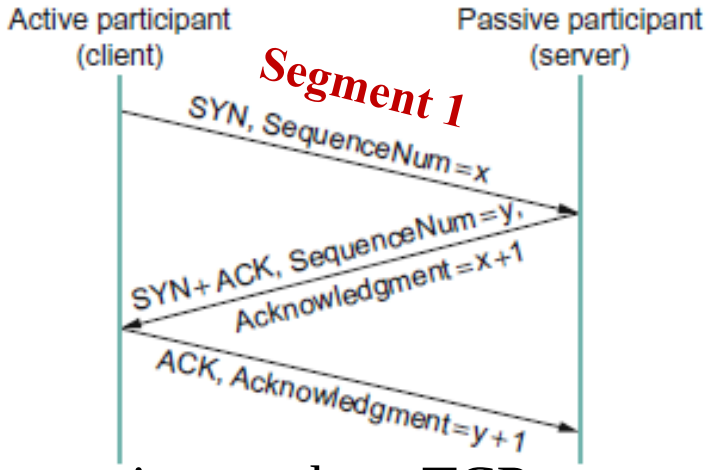
4b. Which means that **SYN** packet is considered to **consume** one byte of data.

5. ACK Acknowledgement from the **Client** with **ACK num = y + 1**

Note: ACK does not consume a Sequence number. It is not required to retransmit an ACK segment if it is lost or not delivered to the intendent recipient.

TCP Connection Establishment - 1

- A connection typically goes through **three phases**:
  1. Setup,
  2. Data transfer (called *established*), an
  3. Teardown (closing).

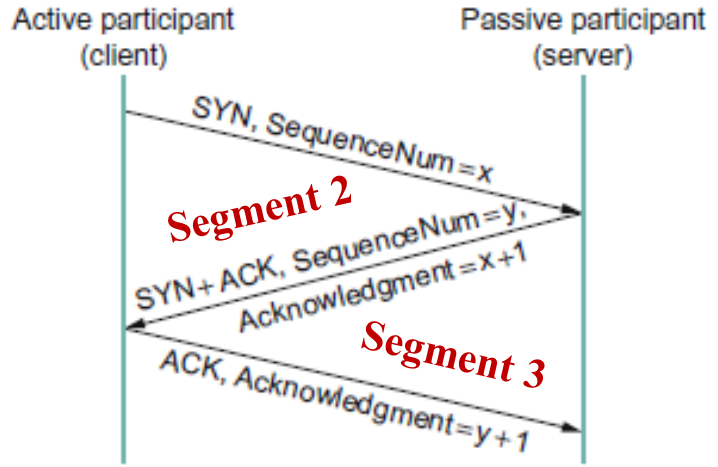


- As we will see, some of the difficulty in creating a robust TCP implementation is handling all of the transitions between and among these phases correctly.
- To establish a TCP connection, the following events usually take place
- 1. The ***active opener*** (normally called the client) sends a SYN **segment 1** (i.e., a TCP/IP packet with the *SYN* bit field turned ON in the TCP header) specifying the port number of the peer to which it wants to connect and the client's **Initial Sequence Number** is **(x)**

TCP Connection Establishment: 2 and 3

2. The server responds with its own SYN segment containing its initial sequence number (**ISN(y)**).

This is the **segment 2**. The server also acknowledges the client's SYN by **ACK'ing ISN(x) plus 1**.



Note that a **SYN consumes one sequence number** and is **retransmitted** if **lost**.

3. The client acknowledges **SYN** from the server by **ACK'ing ISN(y) plus 1**. This is the **segment 3**.

- These three segments complete the connection establishment.
- This is often called the ***three-way handshake***.
- Its main purposes are to let each end of the connection know that a **connection is starting** and the special details that may be carried as **options**, and to **exchange** the **ISNs** of **both ends**.

About Sequence Number and ACK Number

- All bytes in a TCP connection are numbered, beginning at a randomly chosen initial sequence number (ISN).
- The SYN packets consume one sequence number, so actual data will begin at ISN+1.
- The acknowledgement number is the sequence number of the next byte the receiver expects to receive.
- The receiver **ack'ing** sequence number ‘n’ acknowledges receipt of all data bytes less than (but not including) byte number ‘n’.
- The sequence number in the header is always valid, not related to any other control bits in the header.
- Whereas, the acknowledgement number is valid only when the ACK flag is set to one.
- The only time the ACK flag is not set, that is, the only time there is not a valid acknowledgement number in the TCP header, is when the first packet of connection set-up, i.e, SYN packet is sent out.

Quiz 2: Amount of Data Exchanged

Choose the correct option about the total number of bytes exchanged through a socket connection so far from one end to the other, if the segment is having a sequence number currently as **x**.

A. **x** bytes have been sent by the end which is generating this segment.

B. **x+1** bytes have been sent by the end which is generating this segment.

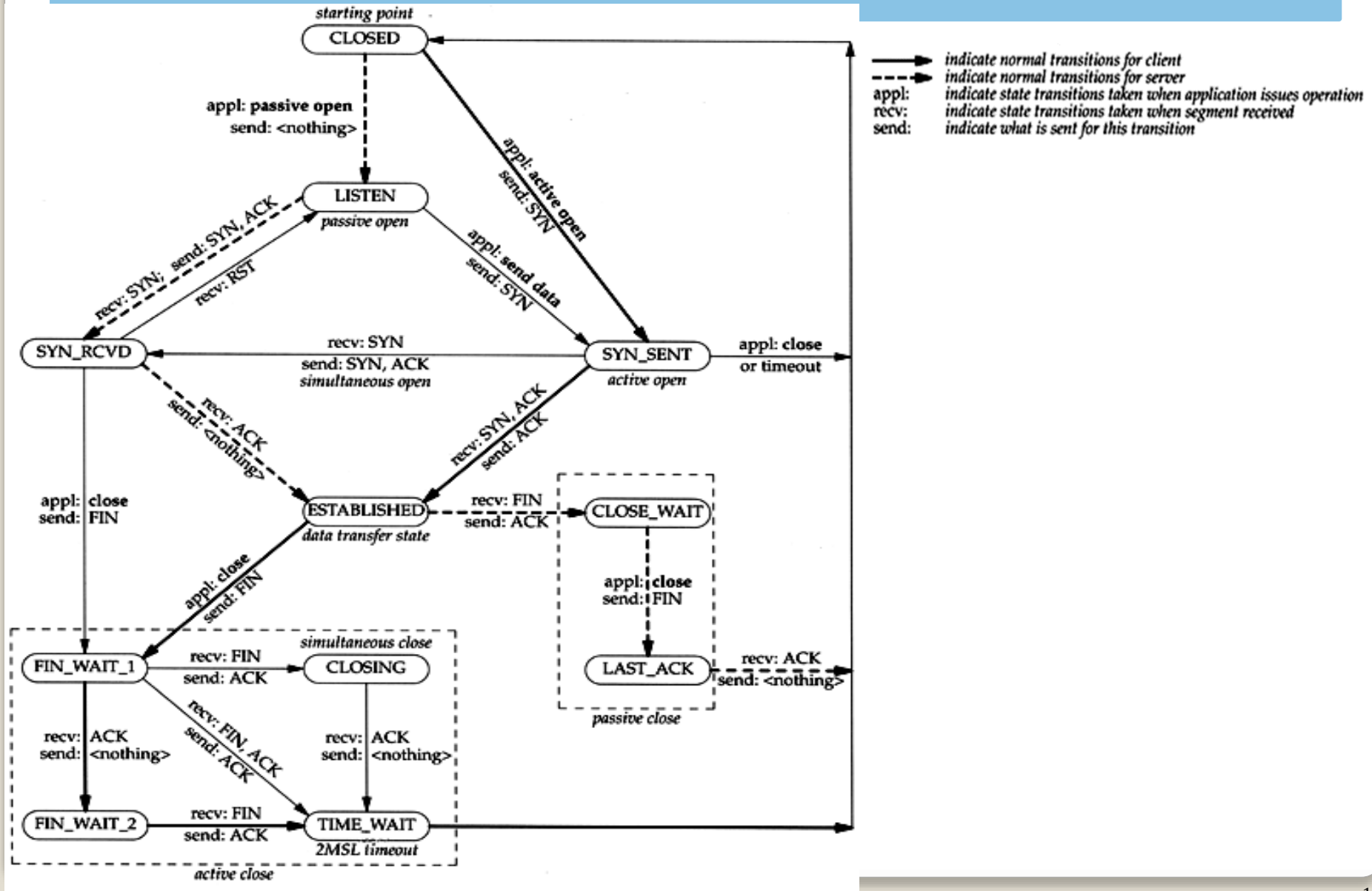
C. **x-1** bytes have been sent by the end which is generating this segment.

D. Nothing can be said about the number of bytes have been exchanged based on the value of sequence number in the segment. **ANS: D**

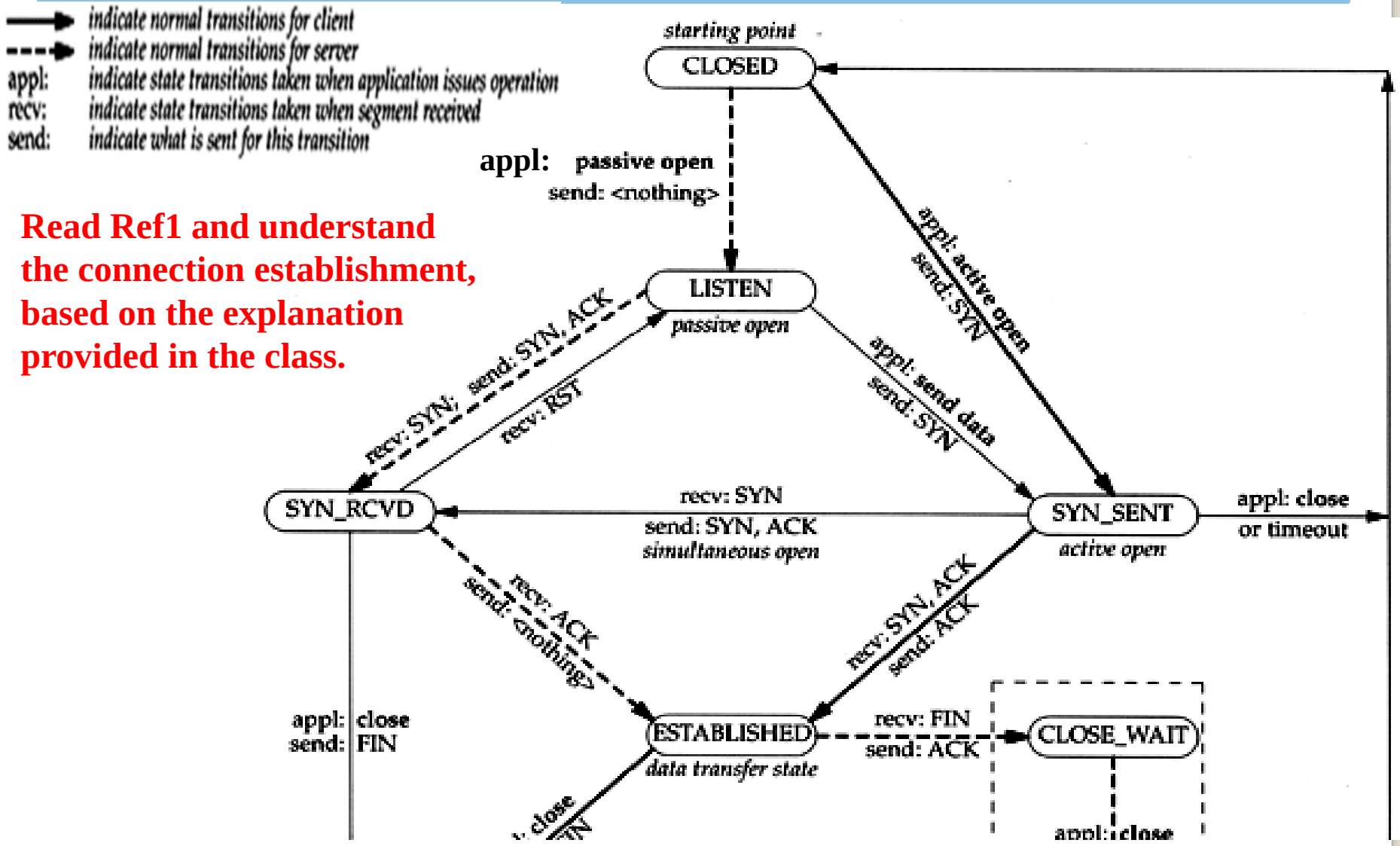
Nothing can be said about the number of bytes exchanged because of the following reasons.

1. The value of **x** does not reveal any information of what was it is ISN (Initial Sequential Number) chosen by the sender.
2. If the ISN is also known, is it possible to find the number of bytes exchanged so far?
   - No, because the sequence number could have wrapped around the max value.

TCP: State Transition Diagram



TCP: Partial State Transition Diagram Shown (connection Setup)



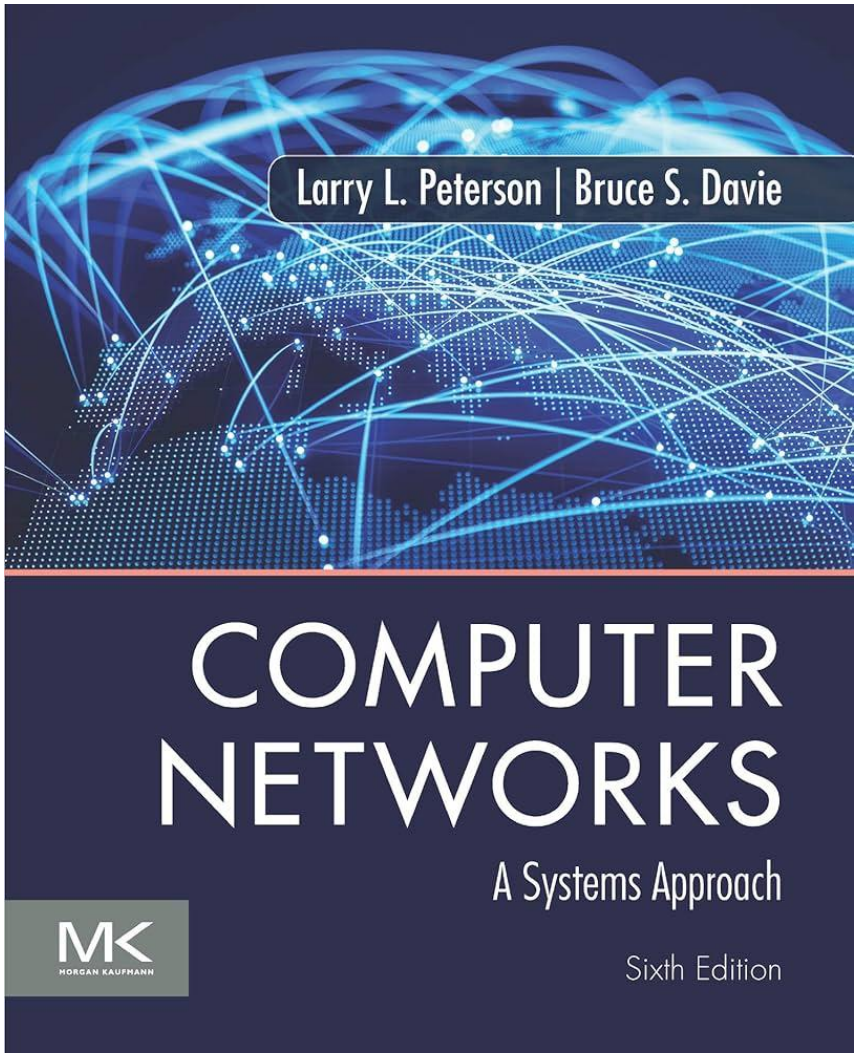
Read Ref1 and understand the connection establishment, based on the explanation provided in the class.

Session 3A: Summary

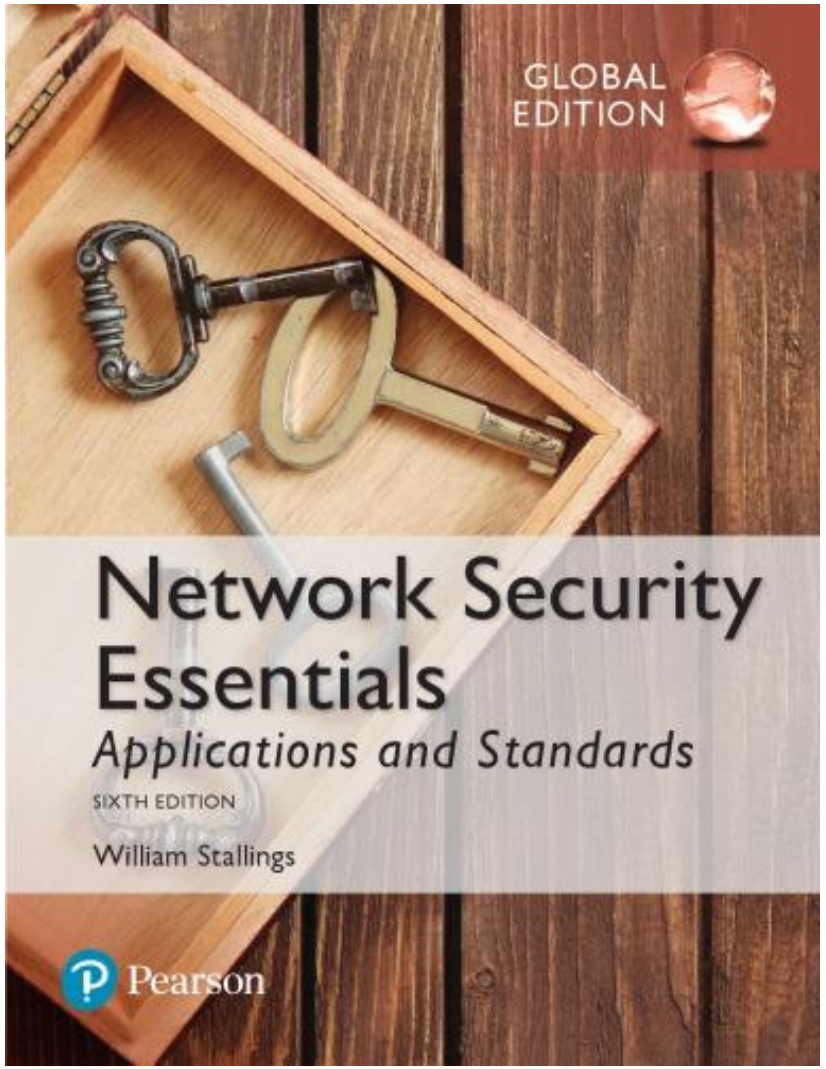
- TCP Connection Management
  - Control Bits: SYN and FIN
  - Quiz 1
- TCP Connection Establishment
  - Three-way handshake
  - Connection Set-up
  - Sequence and Acknowledgement Numbers
- TCP State Transition Diagram
  - Active and Passive Open (client and Server)
  - Simultaneous Open from both machines

Textbooks

Textbook 1

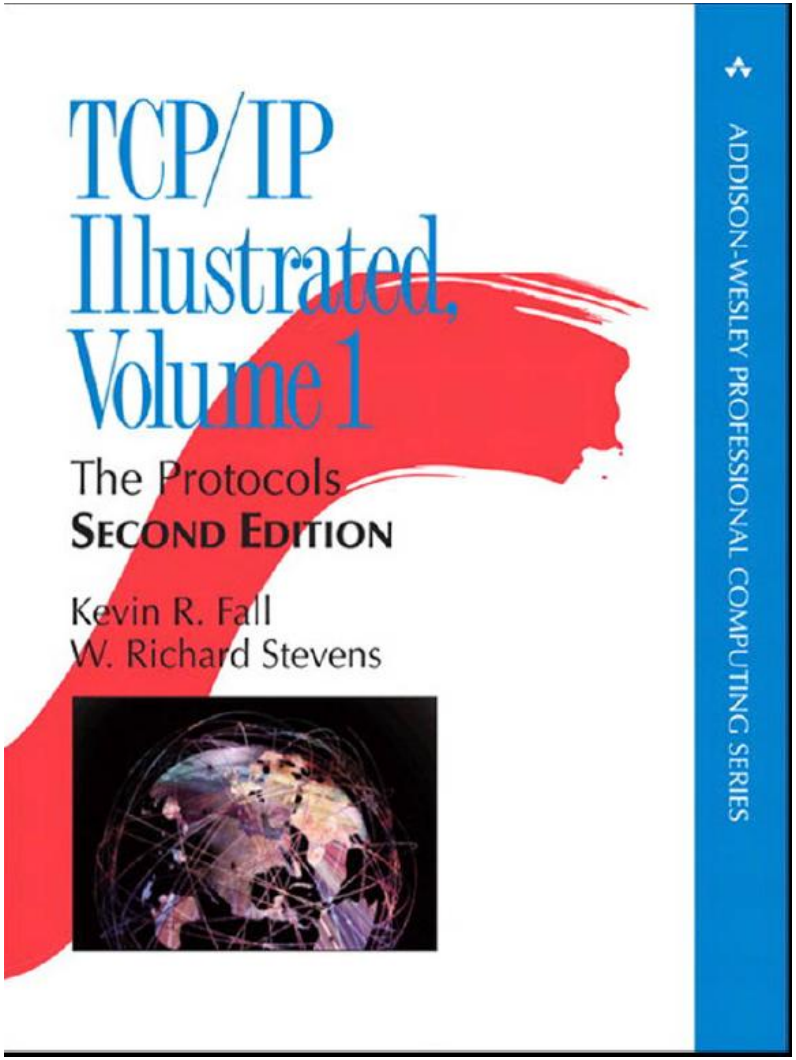


Textbook 2



References

Ref 1



Ref 2

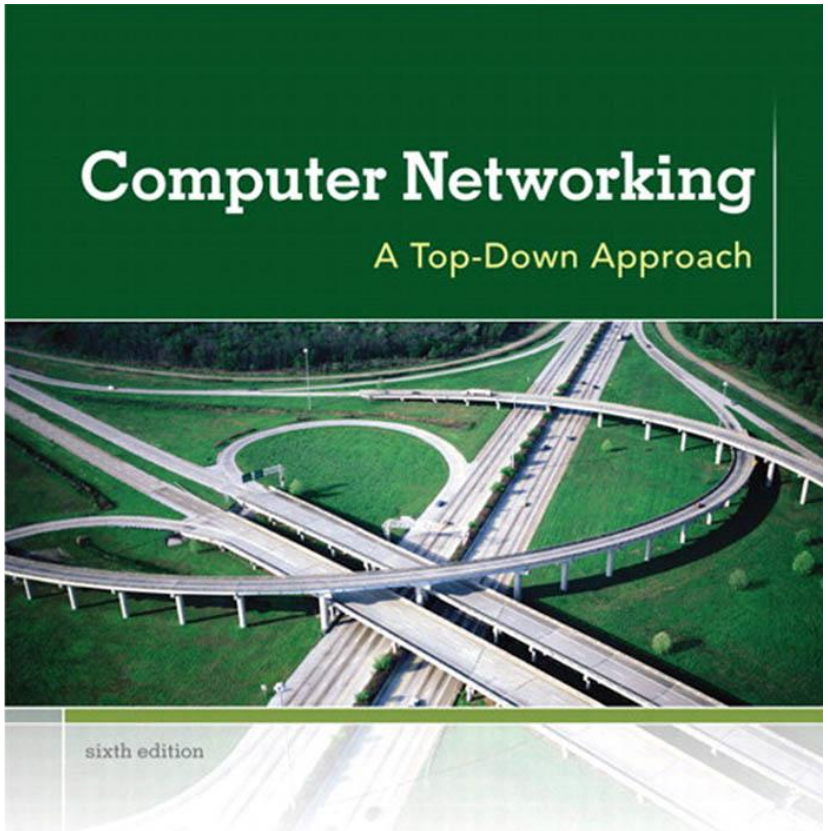
TCP Congestion Control: A Systems Approach

TCP Congestion Control: A Systems Approach

Peterson, Brakmo, and Davie

References

Ref 3



Ref 4