Session 3C

TCP: Half-open, Half-close, Reset

Mouli Sankaran

Session 3C: Focus

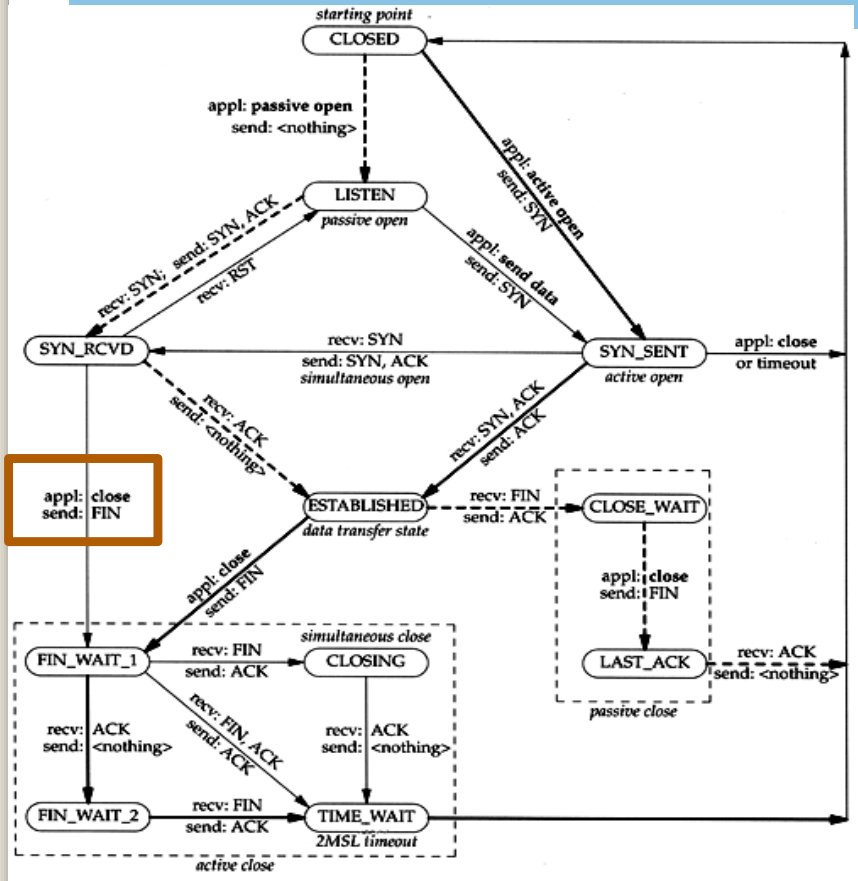
- Half-open condition
- Half-close condition
- Quite time (MSL)
- Reset Segments
- Triggering TCP Data Transmission
  - Three Mechanism

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

Some more Concepts
Based on TCP State Diagram

Quiz 1: Reason for this Transition

Appl: close
Send: FIN

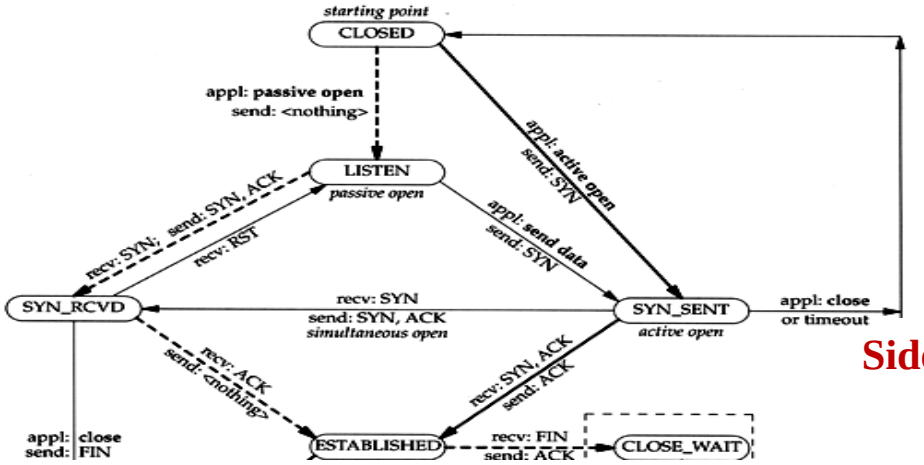


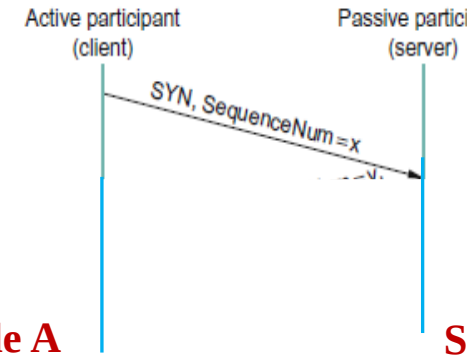
- **Identify what causes this transition.**
- The server application after accepting a connection request from a Client informs the application on its side.
- The Server application can decide to accept or refuse the connection with the client by sending a close message to TCP.
- This will make the state to move from SYN_RCVD to FIN_WAIT1

1. **The potential reasons could be that the server application has reached its limit on number of clients it can be simultaneously connected to. Or**
2. **The server application has black listed the client initiating the request, for some other reasons, so does not want to establish a connection with the client.**

Half-open Condition

Ref: Half-open





Q: If SYN, ACK is not sent by Side B, in response to SYN from Side A, in which state they would be?

Message transactions are shown here

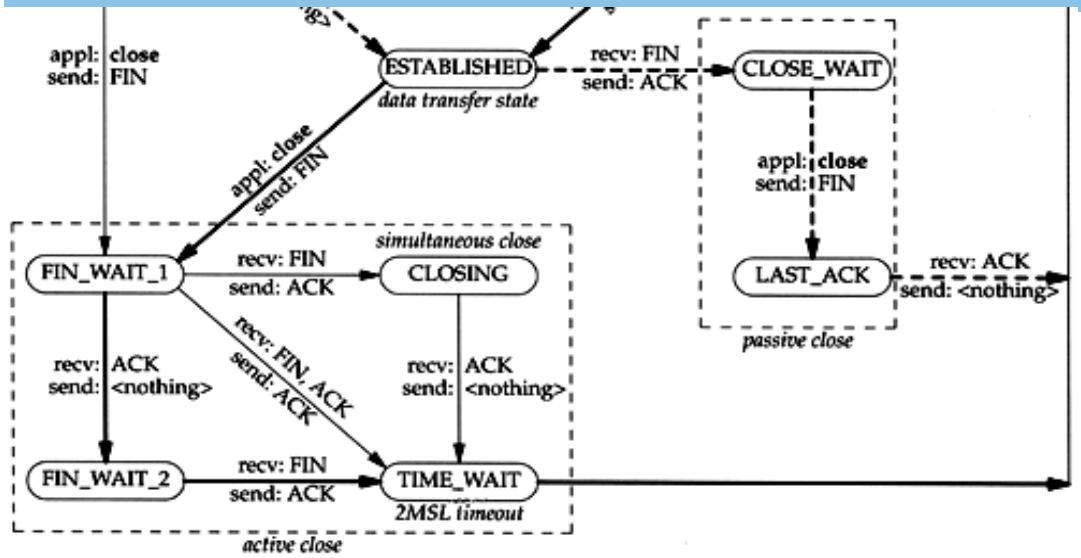
ANS:
Side A (Client): SYN_SENT
Side B (Server): SYN_RCVD

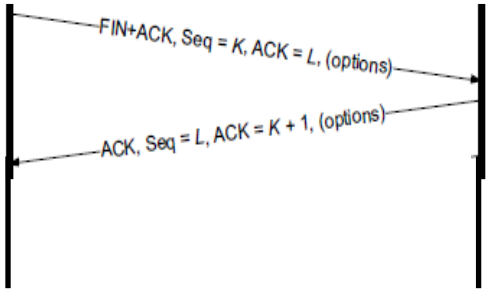
- What is a **Half-open** condition?

1. A connection which is in the process of **being established.** This is **also known as embryonic connection**. It is a **partially created** connection.
2. It **also refers** to TCP connections whose state is **out of synchronization** between the two communicating hosts, possibly due to a crash of one side.

- A TCP connection is referred to as ***half-open*** when the host at one end of that TCP connection has crashed, or has otherwise removed the socket without notifying the other end.
- If the remaining end is idle, the connection may remain in the half-open state for unbounded periods of time.

Half-close Condition





Side A
Active close

Side B
Passive close

Message transactions are shown here

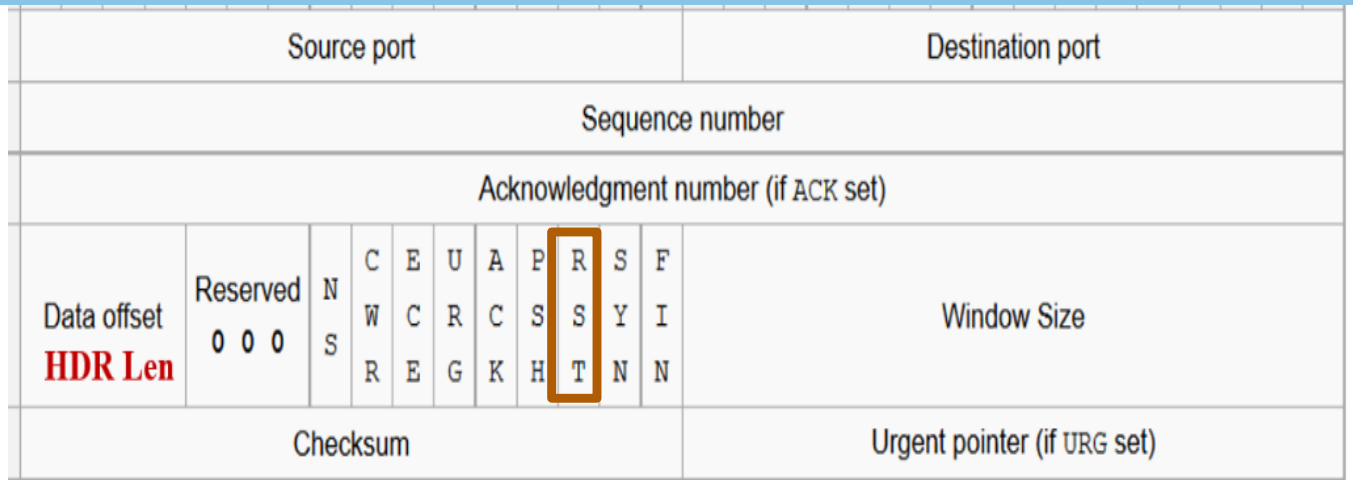
- What is a **Half-close** condition?
- Application on one side has sent a close message and in response to that the TCP has sent a FIN message to the other end and an ACK is also received from the peer
- Which host (Side A or B) could be in **half-close** condition here? **ANS: Both A & B**
- In which state Side A would be while it is in half-close condition?
  - FIN_WAIT_2 state after receiving the ACK from the other end for its FIN message. It is waiting for the other end to close the connection as well.
  - Application on Side B is yet to close the connection, it is in CLOSE_WAIT state.

Quiet Time Concept: While in TIME_WAIT

- 2MSL wait provides a protection against delayed segments from an earlier incarnation of a connection being interpreted as part of a new connection.
- What if a host with connections in the TIME_WAIT state crashes, reboots within the MSL, and immediately establishes new connections using the same local and foreign IP addresses and port numbers corresponding to the local connections that were in the TIME_WAIT state before the crash?
- In this scenario, delayed segments from the connections that existed before the crash can be misinterpreted as belonging to the new connections created after the reboot.
- To protect against this scenario, TCP should wait an **amount of time** equal to the **MSL** before creating any new connections after a reboot or crash. This is called the **quiet time**.

MSL: Maximum Segment Lifetime is the maximum time a TCP segment can exist in a network It is normally set to 2 minutes.

1. Reset: No process waiting on Destination Port



- A segment having this bit set to "ON" is called a "reset segment" or simply a "reset."
- Reset segments do not carry any data but its Sequence number and ACK number (with its ACK bits set) are valid.
- A common case for generating a reset segment is when a connection request arrives (SYN message from the Client) and **no process** is **listening** on the **destination port** at the *Server*.

2. Reset: Incorrect Segment Received

- In general, a **reset is also sent** by TCP whenever a **segment arrives** that **does not appear to be correct** for the **referenced connection**.
  - The term referenced connection means the connection specified by the **4-tuple** in the **TCP and IP headers** of the **segment**.
- Resets ordinarily result in a fast teardown of a TCP connection.
- How to **validate** a **Reset segment** received?
- For a reset segment to be accepted by a TCP, the ACK bit field must be set and the ACK Number field must be within the valid window of the sender of the Reset segment.
- This helps to prevent a simple attack in which any intruder who is able to generate a reset, matching the appropriate connection (4-tuple) could disrupt a connection in progress.

3. Reset: To Abort a Connection

- We know that a normal way to terminate a connection is for one side to send a FIN.
- This is sometimes called an **orderly release** because the FIN is sent after all previously queued data has been sent, and there is normally no loss of data.
- But it is also possible to **abort a connection** by **sending** a **reset instead of a FIN** at **any time**. (that is why it needs to be validated)
- This is sometimes called an **abortive release**.
- **Aborting a connection** provides **two features** to the **application**:

1. Any queued data is thrown away and a reset segment is sent immediately, and
2. The receiver of the reset can tell that the other end did an abort instead of a normal close to the application on its side.

Triggering Transmission

Reference: Ref1: Computer Networks A Systems Approach:
Section 5.2.5: Triggering Transmission

Triggering Transmission

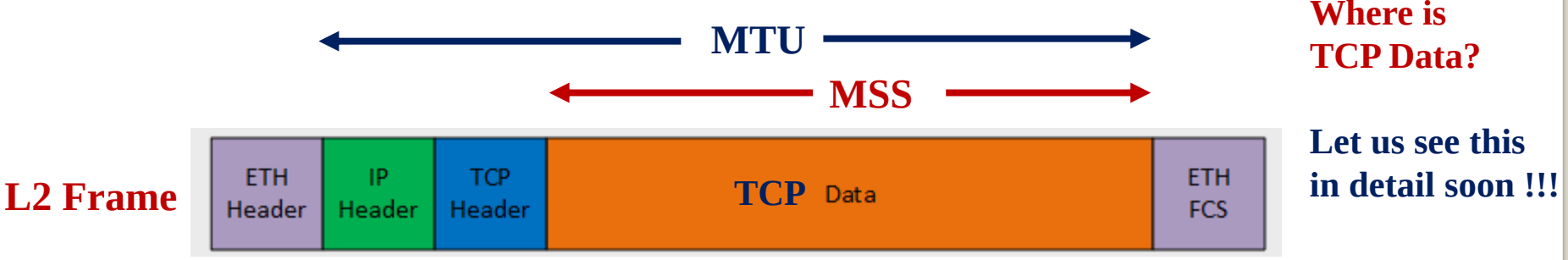
- We next consider a surprisingly **subtle** issue: how TCP decides to transmit a segment.
- As described earlier, TCP supports a byte-stream abstraction; that is, application programs write bytes into the stream, and it is up to TCP to decide that it has enough bytes to send a segment.
- What **factors** govern **this decision**?
- If we **ignore** the possibility of **flow control**, that is, we assume the **window** is **wide open**, as would be the case when a **connection first starts**.
- Then, TCP has **three mechanisms** to **trigger** the **transmission** of a **segment**.

Note: **Window** here **refers to** the available **window size** as per "**Sliding window**" algorithm.

Meaning of Subtle: (especially of a change or distinction) so delicate or precise as to be difficult to analyze or describe.

First Mechanism: Triggering Transmission

- TCP maintains a variable, typically called the **Maximum Segment Size** (**MSS**), and it sends a segment as soon as it has collected MSS bytes from the sending process.
- MSS is usually set to the size of the largest segment TCP can send **without causing the local IP to fragment**. ← **Why should it be avoided?**



Where is TCP Data?

Let us see this in detail soon !!!

- That is, MSS is set to the maximum transmission unit (MTU) of the directly connected network (datalink layer), minus the size of the TCP and IP headers
  - MSS = MTU of connected network – (Size of IP header + TCP header)
  - As you are aware, the TCP and IP header sizes are 20 bytes each, totaling 40 bytes of overhead for a TCP segment, with no optional fields added in both.

Second Mechanism: Triggering Transmission

- The second trigger is when the sending process explicitly asks TCP to send the data that has already been buffered in its Tx buf.
- As you are aware of the control flag PUSH that TCP supports for a PUSH operation, which makes the TCP stack to send whatever data available, without waiting for more data from the sender application.
- TCP can send all the data that it has, as long as the current window size advertised by the receiver is higher than the data that needs to be sent.

Third Mechanism: Triggering Transmission

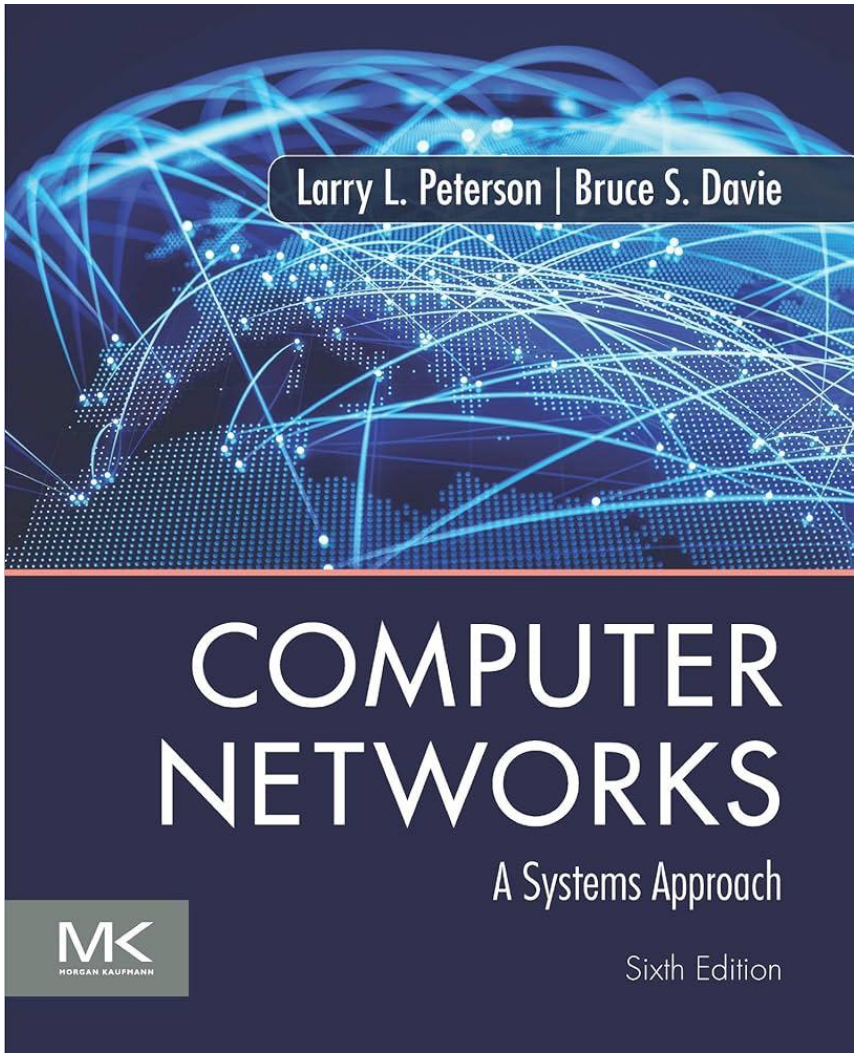
- The **final trigger** for transmitting a segment is that **a timer fires**; the resulting segment that is sent would contain as many bytes as there are currently in the Tx buf for transmission.
- Time value is decided based on the RTT value of the connection between the hosts and the kind of applications requirement, interactive or data transfer, etc.
- Timer is set to RTT, to allow ACK to be received for the data sent. If the timer fires, that means either the data sent to the other end was lost or the ACK from the peer was lost.
- This result in a retransmission of data.

Session 3C: Summary

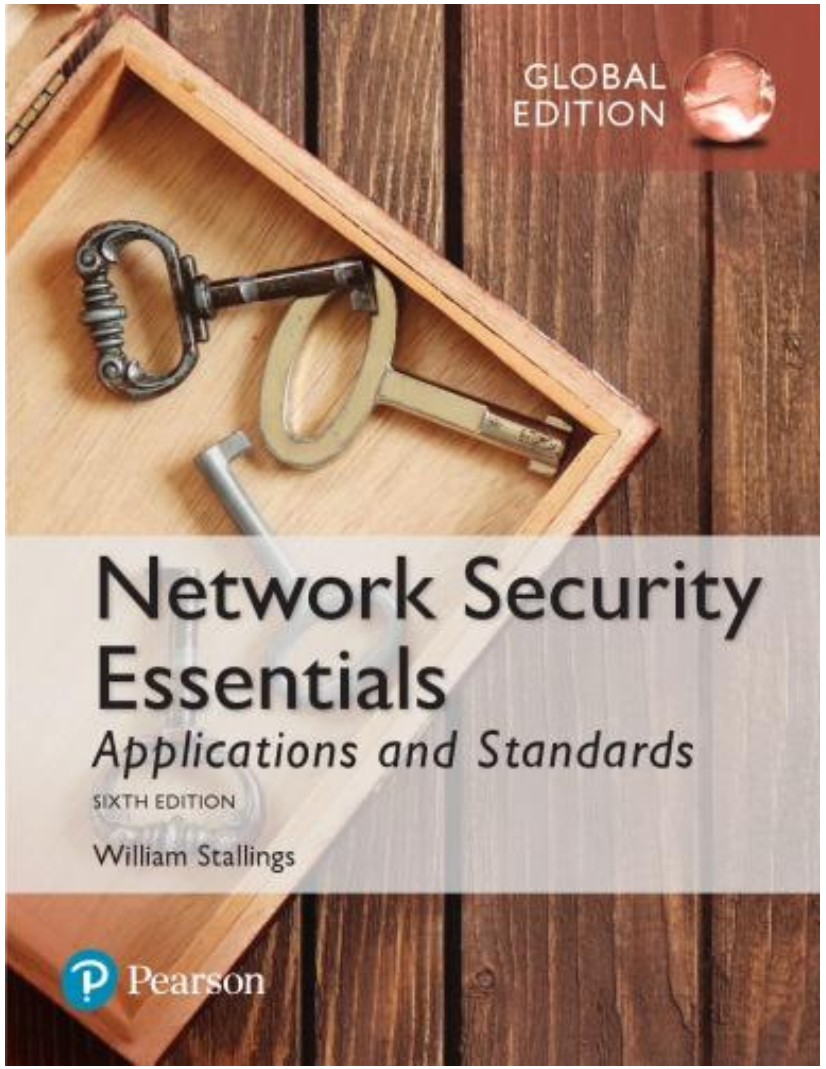
- Half-open condition
- Half-close condition
- Quite time (MSL)
- Reset Segments
- Triggering TCP Data Transmission
  - Three Mechanism

Textbooks

Textbook 1

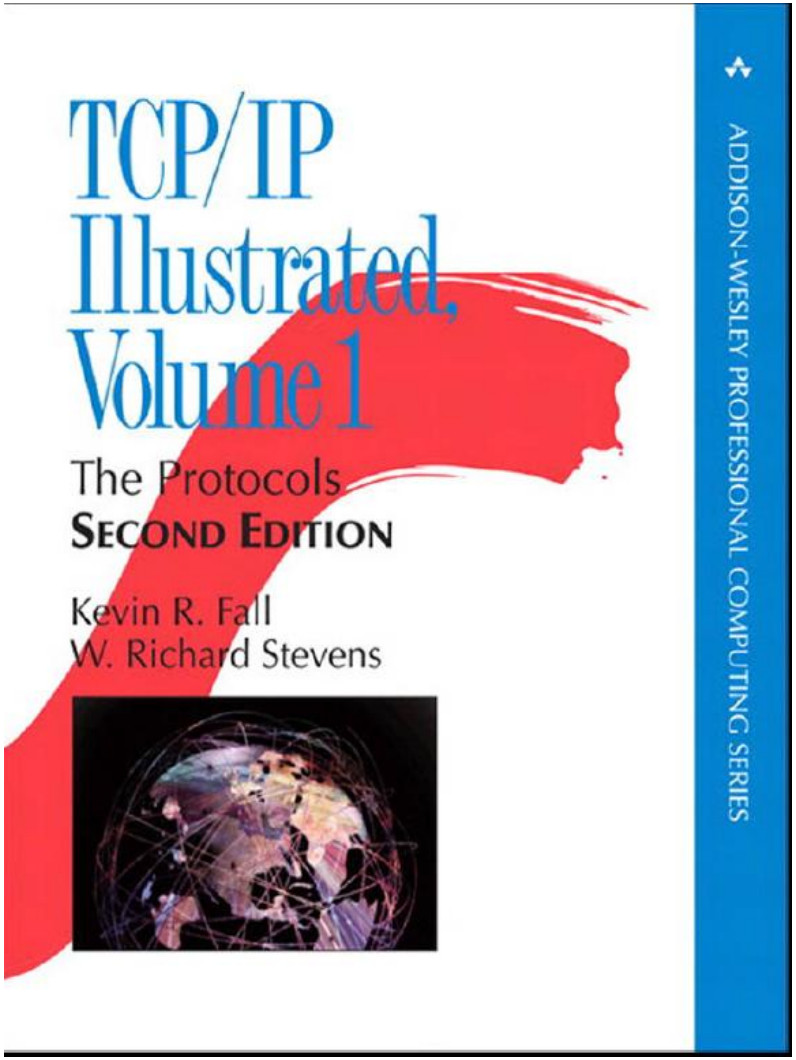


Textbook 2



References

Ref 1



Ref 2

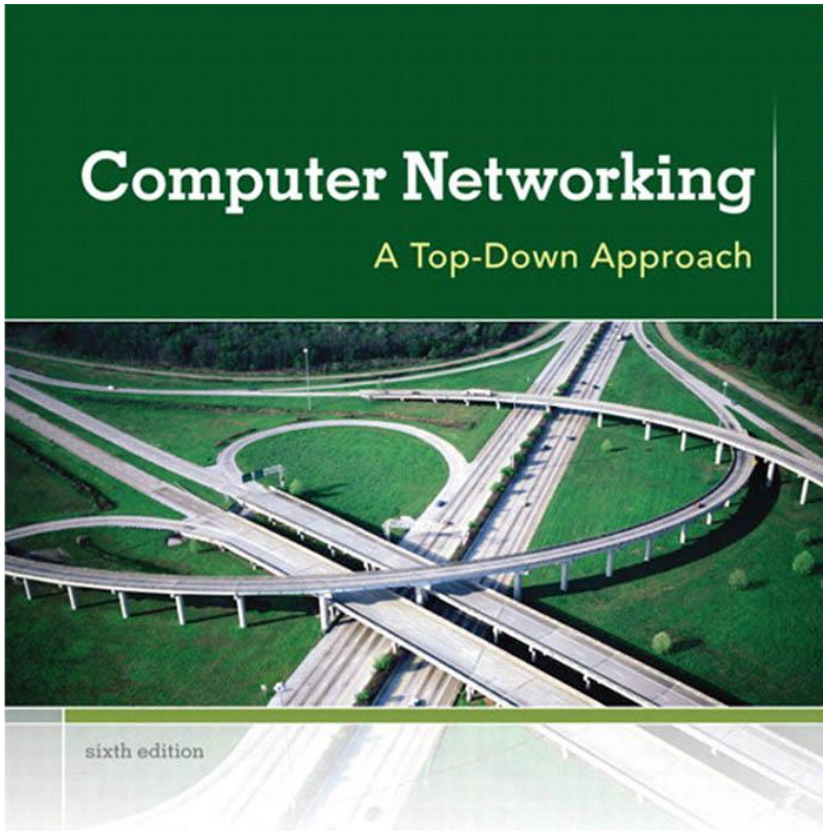
TCP Congestion Control: A Systems Approach

TCP Congestion Control: A Systems Approach

Peterson, Brakmo, and Davie

References

Ref 3



Ref 4