Session 2E

TCP Features

Mouli Sankaran

Session 2E: Focus

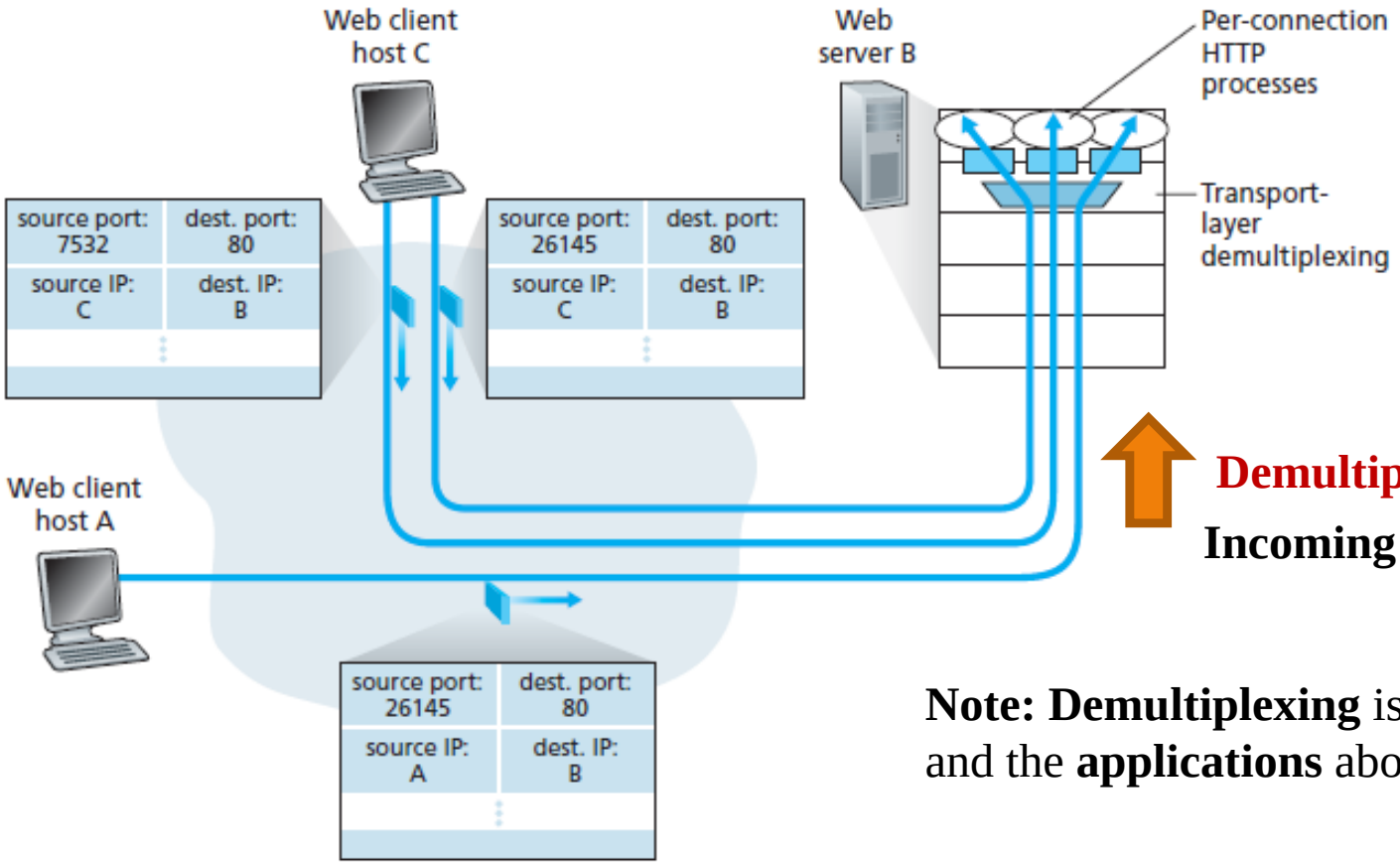
TCP Features:

- Demultiplexing
- Multiplexing
- TCP segment without any Data
- Tx Buf and Rx Buf
  - Circular Buffers
- Reliable TCP connection using Unreliable IP
  1. Packet loss (data)
  2. Packet loss (ACK)
  3. Long delay in Packet delivery
  - Solutions from TCP for the above three issues

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

TCP Features

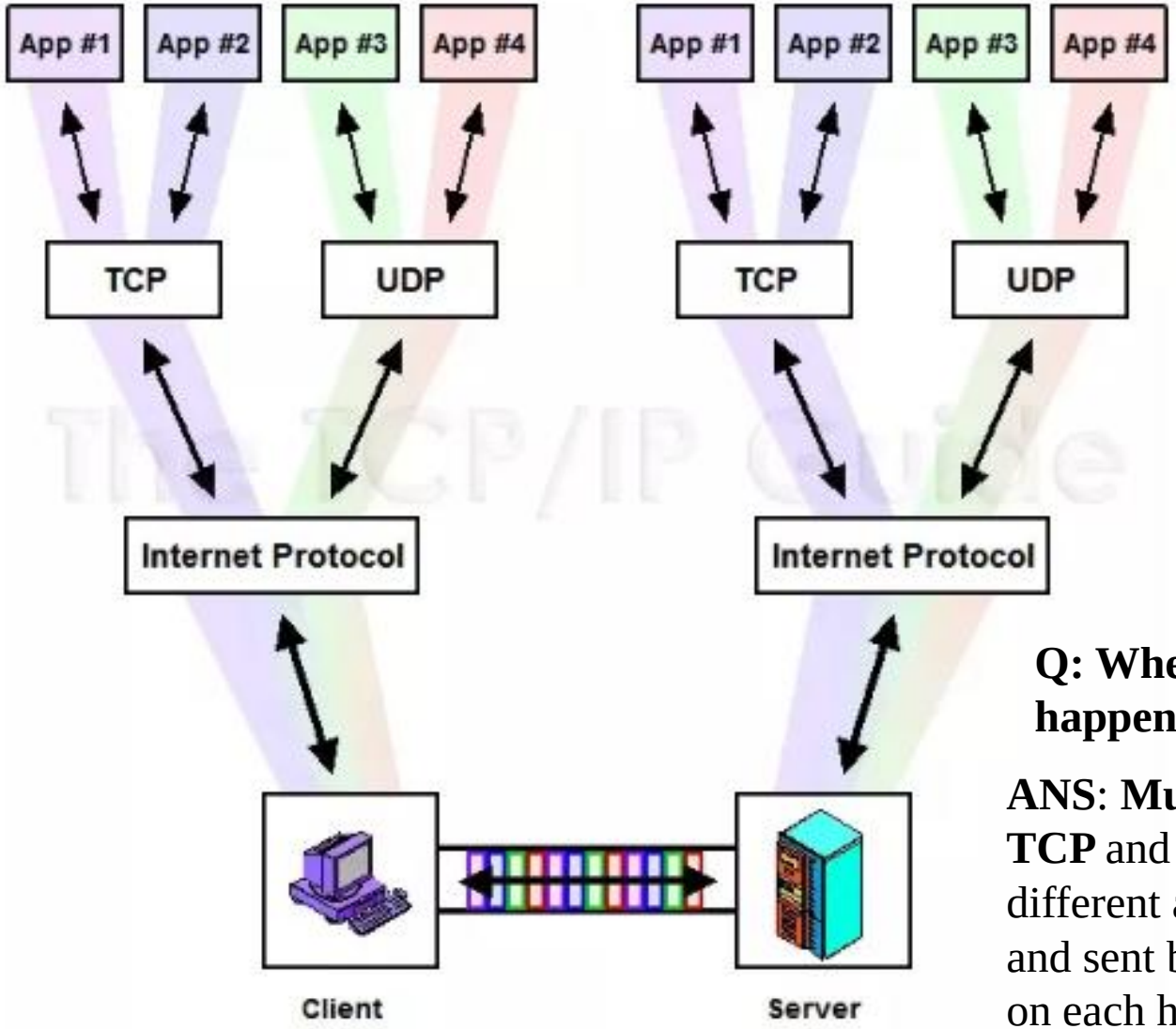
TCP: Demultiplexing



Note: Demultiplexing is between **TCP** and the **applications** above it

- The server host may support many simultaneous TCP connection sockets, with each socket attached to a process, and with each socket identified by its own four tuple.
- When a TCP segment arrives at the host, all four fields (source IP address, source port, destination IP address, destination port) are used to direct (**demultiplex**) the segment to the appropriate socket

TCP: Multiplexing



Q: Where does **Multiplexing happen** here?

ANS: **Multiplexing** happens between **TCP** and the **IP layer**. Segments from different applications are **multiplexed** and sent by a single IP layer running on each host

TCP Segment without any Data

- Suppose the receiver (host) of a TCP data segment does not have any data to send to the sender (host), the receiver is still should ACK the data received, so that the sender can keep sending the data.
- In this scenario, the receiver of the TCP data segment sends a TCP header with **ACK bit set** and **Acknowledgement number** of the next data it is expecting from the sender.
- There is no data sent along with the segment from the receiver.
- The length in the IP header is adjusted accordingly.
- Note that the sequence number is not advanced, because no data is being sent here.

An ACK segment, if carrying no data, consumes no sequence number.

Quiz 1: Choose all the correct Options

- Why should the receiver of TCP data segments should ACK even if it does not have any data to be sent to the other end?

A. The receiver's Rx buffer will overflow if ACK is not sent to the sender

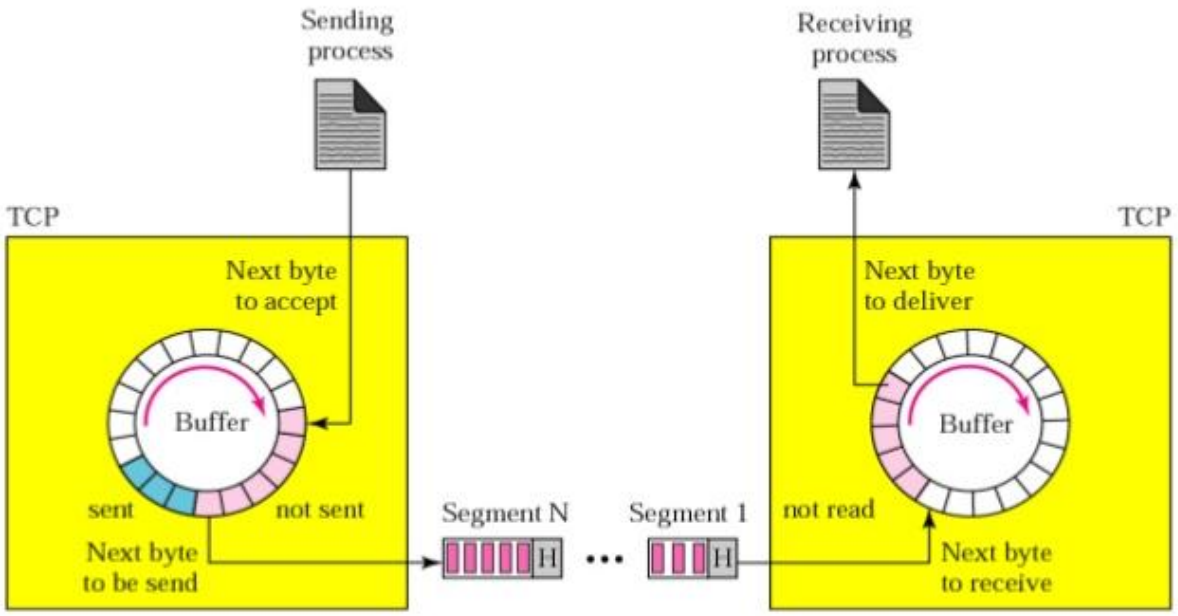
B. The sender's Tx buffer cannot be emptied or cleared if no ACK is received from the Rx side

C. The sender cannot take more data from the application, if it does not get ACKs from the receiver and it empties its Tx buf

D. If ACK is not received by the sender, timeout will happen on the sender and it will start resending the data again which is a waste of network bandwidth and time loss.

ANS: B, C and D

TCP: Tx Buf and Rx Buf



Both the Tx and Rx buffers are **circular buffers**, with read and Write pointers.

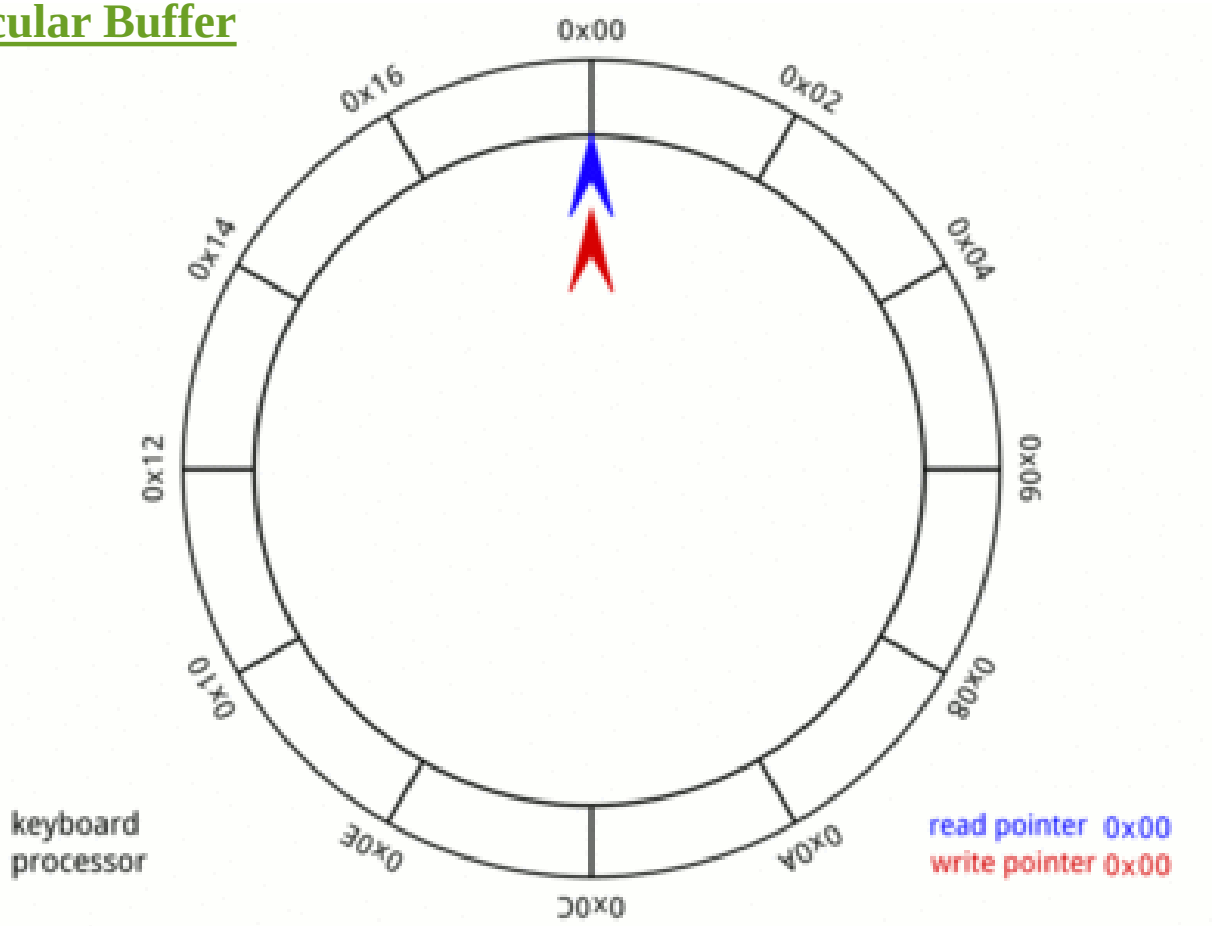
A **circular buffer**, **circular queue**, **cyclic buffer** or **ring buffer** is a data structure that uses a single, fixed-size buffer as if it were connected end-to-end.

This structure lends itself easily to buffering data streams

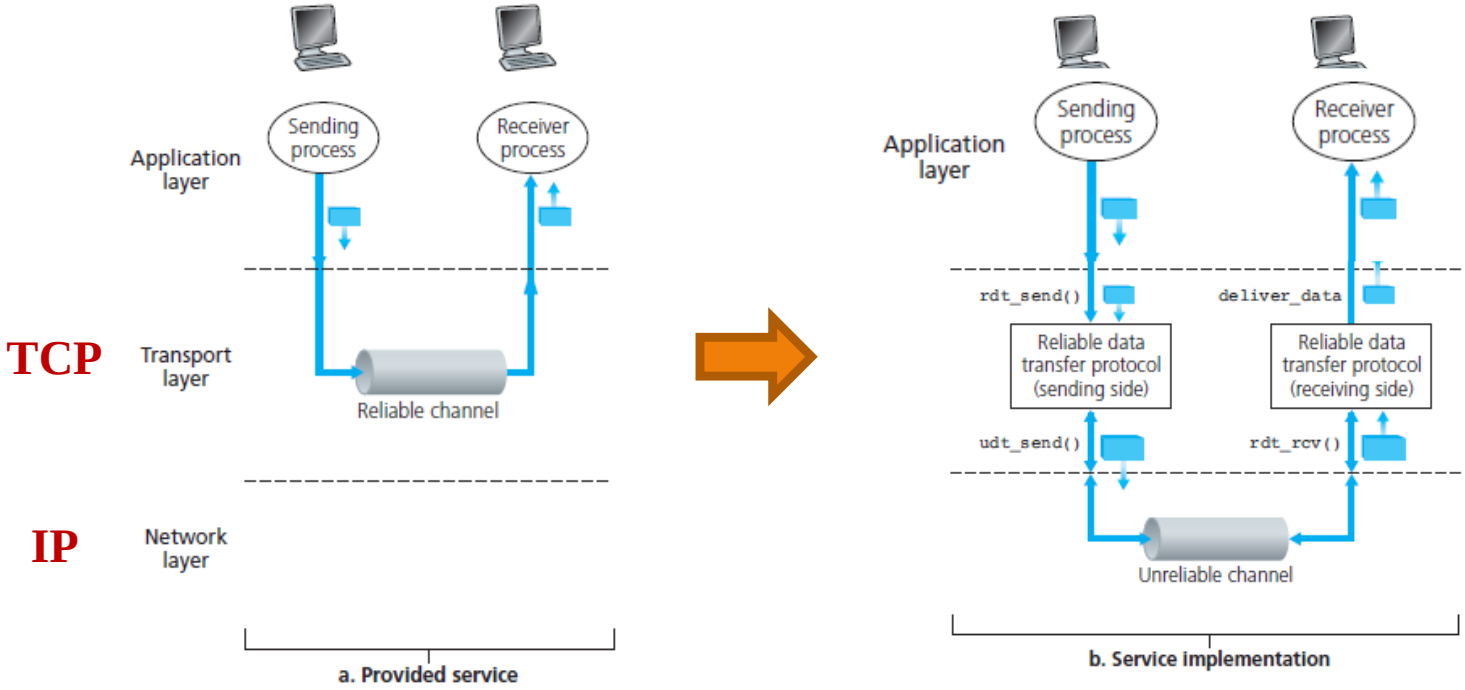
- Since TCP is a byte streaming protocol and every bytes sent is to be acknowledged, the buffers maintained by the Tx and Rx need to be able to handle these requirements.
- Remember that TCP does not do any marking of data, unless some urgent data is to be sent to the other end.
- As the ACKs are received the Tx buffers are advanced, creating space for the writing of data by the sender process.

Example: Circular Buffer

Ref: Circular Buffer



Reliable (TCP) using Unreliable (IP) Protocol

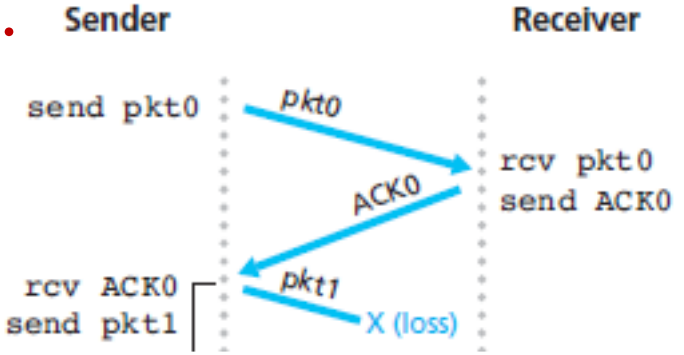


- "**Service provided**" by the TCP to the application layer above is to give a reliable channel between the two ends, using the unreliable protocol underneath (IP).
- It is the **responsibility** of a **reliable data transfer protocol (TCP)** to **implement** this **service abstraction**.

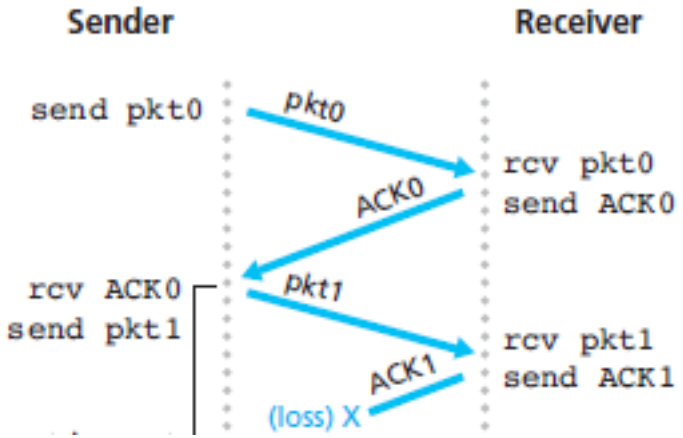
Issues with Unreliable Channel

1. **Packet loss (data)**
2. **Packet loss (ACK)**
3. **Long delay in packet delivery**

1.

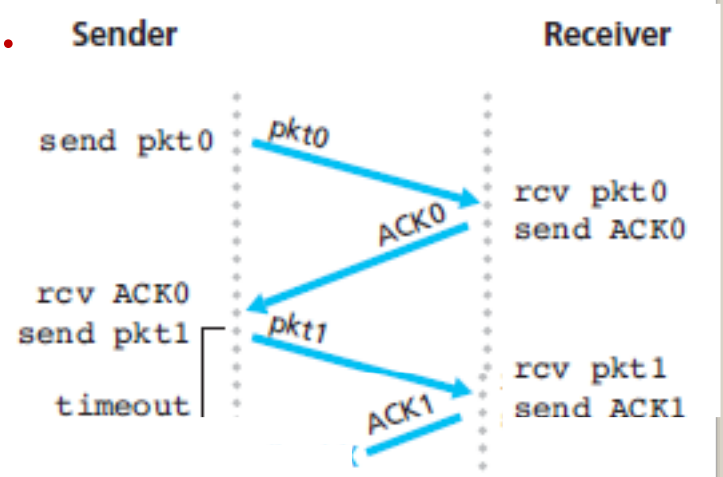


2.



Long delay in Receiving the ACK from Rx

3.



Q1: These issues are with which protocol layer?

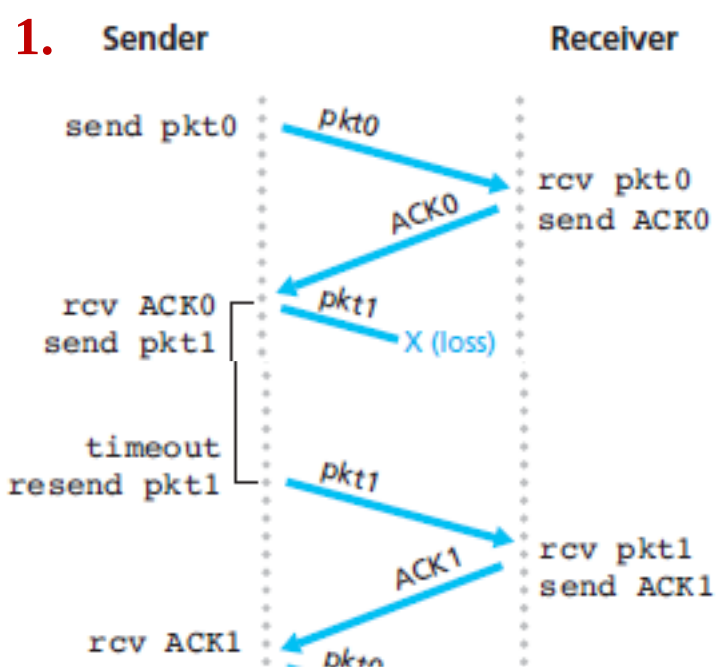
ANS: IP (Network Layer)

What are these above **diagrams** called as?

Protocol Sequence diagrams or Event Diagrams

TCP Handling: 1. Packet loss (data)

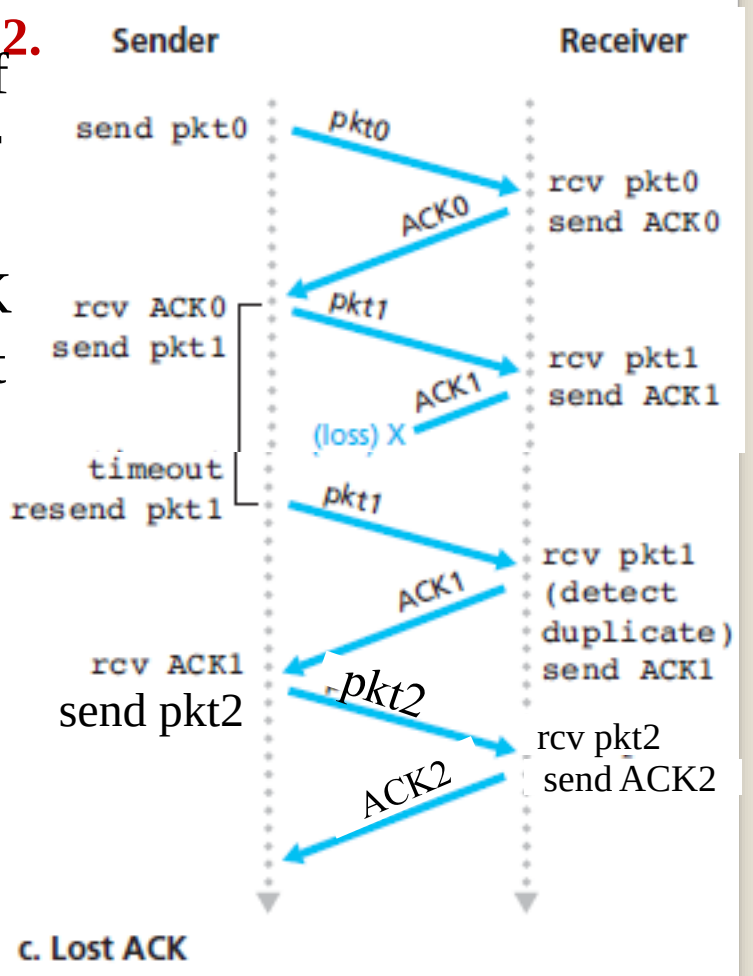
- Sender (TCP) starts a timer for each TCP segment sent over the network
- If an ACK for that particular segment is not received and the timer expires, the sender sends the same TCP segment again
- If the ACK is received for a particular single segment or a collective ACK for a set of segments, the relevant timers are switched off, and thus no timeout happens, and so, no re-transmission of that segment as well.



TCP Handling: 2. Packet loss (ACK)

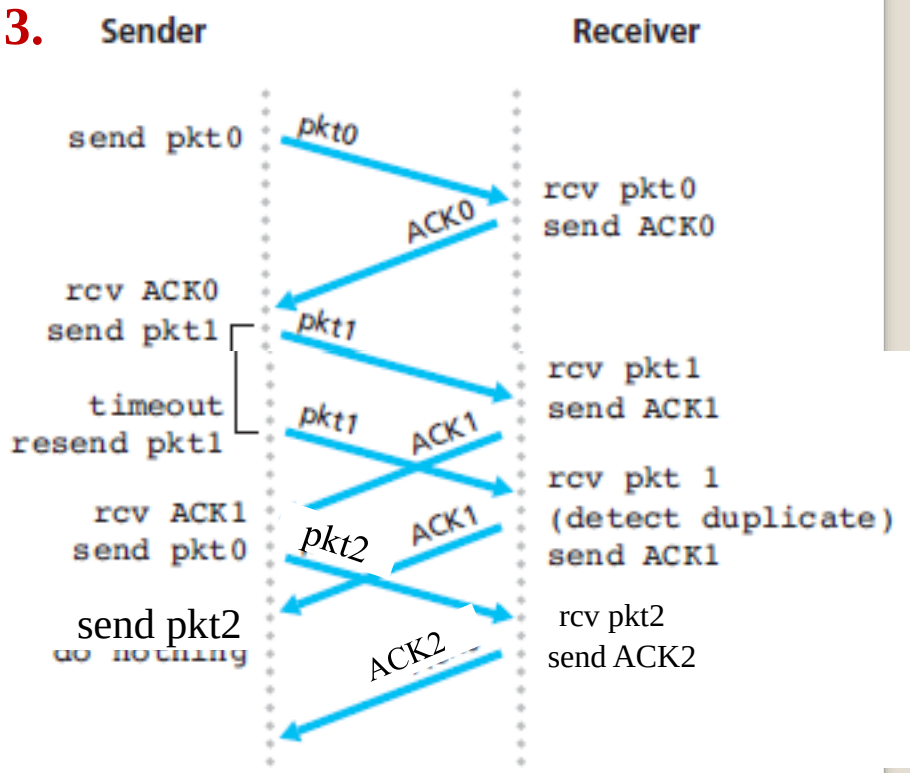
- When the sender does not receive the ACK for a data segment, the sender has no way of finding whether the data segment got lost or the ACK for that data segment got lost.
- Similarly, the end which is sending the ACK has no way of finding whether the ACK that was sent by it was received by the other end or not.
- So, the only way to resolve this issues is by sending the TCP segment again.
- When the receiver gets a duplicate segment it understands that it's earlier ACK was not received by the sender and
- And thus, it again acknowledges the duplicate segment, though it has that data already in its Rx buffer.

2.



TCP Handling: 3. Long delay Packet delivery

- When the ACK for a data segment sent by the receiver takes more time to reach the sender, the sender assumes that either the data segment sent was lost or the ACK for the segment from the receiver was lost.
- The only way to resolve this issue is for the sender to re-send the same segment again, after the timeout.
- Now, the delayed ACK gets received, thus the sender proceeds with the sending of next data.



Note: This will cause duplicate data Packets and ACKs to be received. But the duplicate Data packets and ACKs are ignored.

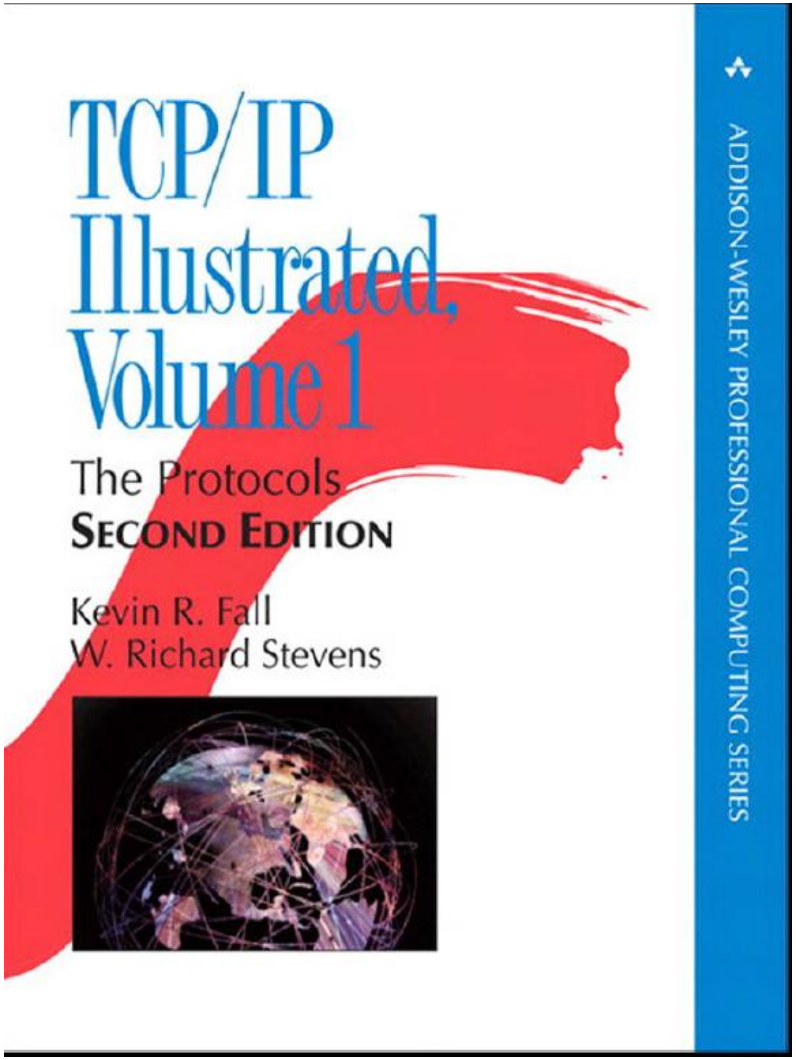
Session 2E: Summary

TCP Features:

- Demultiplexing
- Multiplexing
- TCP segment without any Data
- Tx Buf and Rx Buf
  - Circular Buffers
- Reliable TCP connection using Unreliable IP
  1. Packet loss (data)
  2. Packet loss (ACK)
  3. Long delay in Packet delivery
  - Solutions from TCP for the above three issues

References

Ref 1



Ref 2

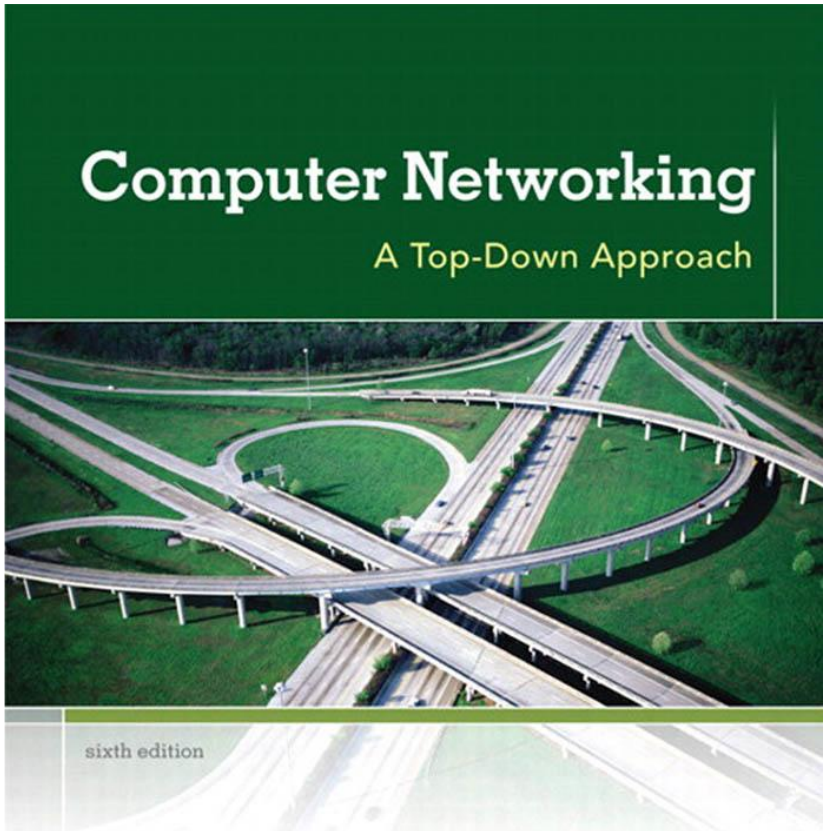
TCP Congestion Control: A Systems Approach

TCP Congestion Control: A Systems Approach

Peterson, Brakmo, and Davie

References

Ref 3



Ref 4