## Week 4

### 1. Reliable Data Transfer (RDT 3.0 Recap)

* RDT 3.0 improves reliability by handling packet loss and corruption.
* Uses ACKs (Acknowledgments) and timeouts to detect lost packets.
* Uses a stop-and-wait approach, meaning the sender waits for an ACK before sending the next packet.
* Problems with Stop-and-Wait

1. Low efficiency – Only one packet is sent at a time.
2. Long waiting time – Acknowledgment delays slow down data transmission.

### 2. Pipelining and Performance Improvement

To improve efficiency, pipelining is introduced, allowing multiple packets to be in transit at once. This requires:

* Larger sequence numbers (to track multiple packets).
* Buffering at sender/receiver (to handle out-of-order packets).

Two Types of Pipelined Protocols

1. Go-Back-N (GBN)

* Sender can send N unacknowledged packets.
* If a packet is lost, the sender retransmits all packets from the lost one onwards.
* Receiver only acknowledges the last correctly received packet.

1. Selective Repeat (SR)

* Sender retransmits only the lost packet instead of all packets.
* Receiver buffers out-of-order packets until missing ones arrive.
* More efficient but complex due to additional memory usage.

### 3. TCP (Transmission Control Protocol)

TCP is a reliable, connection-oriented transport protocol used in most network applications.

Key Features of TCP

* Reliable Data Transfer – Ensures all data is received correctly.
* Flow Control – Prevents sender from overwhelming the receiver.
* Congestion Control – Prevents overloading the network.
* Connection-Oriented – Uses a three-way handshake for connection establishment.
* Cumulative Acknowledgments – Acknowledges all received data up to a certain point.

### 4. TCP Segment Structure

A TCP segment consists of:

* Source & Destination Port – Identify applications.
* Sequence Number – Helps order packets correctly.
* Acknowledgment Number – Confirms received data.
* Flags (e.g., SYN, ACK, FIN) – Controls connection setup/teardown.
* Receive Window – Used for flow control.
* Checksum – Provides error detection.

### 5. TCP Connection Management

TCP uses a three-way handshake for connection setup:

1. Client sends SYN (synchronization request).
2. Server responds with SYN-ACK (acknowledgment).
3. Client sends ACK, and the connection is established.

Connection Termination

* Uses FIN-ACK-FIN-ACK exchange to close the connection gracefully.

### 6. TCP Retransmission and Timeout

* TCP ensures reliability by retransmitting lost packets using timeouts and ACKs.
* How TCP Handles Retransmissions:

1. Timeout-Based Retransmission

* If an ACK is not received in time, TCP retransmits the segment.
* Uses an adaptive timeout based on estimated round-trip time (RTT).

1. Fast Retransmit

* If three duplicate ACKs are received, TCP retransmits the missing segment immediately (instead of waiting for a timeout).

### 7. TCP Flow Control

Flow control ensures the sender does not overwhelm the receiver.

* TCP uses a receiver window (rwnd) in the header to inform the sender how much data it can handle.
* If the receiver’s buffer is full, it reduces the window size to slow down the sender.

### 8. TCP Congestion Control

Congestion occurs when too many packets are sent, causing packet loss and delay.

Key TCP Congestion Control Mechanisms

1. Slow Start

* TCP starts with a small window size and increases exponentially until packet loss occurs.

1. Congestion Avoidance

* Once a threshold is reached, TCP increases the window size linearly to avoid congestion.

1. Multiplicative Decrease

* When a packet loss is detected, TCP cuts the window size by half.

1. Fast Recovery

* Instead of dropping to the initial slow start, TCP reduces congestion gradually after detecting multiple duplicate ACKs.

### 9. Advanced TCP Congestion Control Algorithms

TCP CUBIC

* Improves over standard TCP AIMD (Additive Increase Multiplicative Decrease).
* Increases the window size faster when far from congestion and slows down near maximum throughput.
* Used by Linux and modern internet applications.

Delay-Based Congestion Control

* Monitors RTT changes to detect congestion before packet loss occurs.
* Example: BBR (Bottleneck Bandwidth and RTT) used by Google.

### 10. Explicit Congestion Notification (ECN)

* A router marks packets instead of dropping them to indicate congestion.
* The receiver then notifies the sender, reducing the sending rate without causing retransmissions.

### 11. TCP Fairness

* Fairness ensures multiple TCP connections share bandwidth equally.
* Issues with fairness:
  + UDP-based applications (e.g., video streaming) don’t use congestion control, leading to unfair bandwidth usage.
  + Parallel TCP connections (used by browsers) can increase data transfer rates unfairly.

### 12. Evolution of Transport Layer – QUIC

QUIC (Quick UDP Internet Connections) is an advanced protocol that improves HTTP/3 performance over UDP.

Features of QUIC

* Faster connection establishment – Combines encryption and handshake in one RTT.
* Multiplexing – Supports multiple streams without head-of-line blocking.
* Better congestion control – Uses modern techniques like BBR.

### 13. Summary

* TCP ensures reliable, ordered communication with congestion and flow control.
* Go-Back-N and Selective Repeat improve efficiency in reliable transfer.
* Congestion Control (TCP AIMD, CUBIC, ECN) prevents network overload.
* New protocols like QUIC improve transport-layer efficiency over UDP.