Week4: Transport Layer Pt2

NOS_Lecture_slides_WK04.pdf

Transport Layer Summary

Reliable Data Transfer (rdt3.0)

- **Stop-and-Wait Protocol**: Sender waits for an acknowledgment (ACK) for each packet before sending the next.
- **Performance**: Low utilization due to waiting for ACKs.
- **Pipelining**: Increases utilization by sending multiple packets before receiving ACKs.

Go-Back-N Protocol

- **Sender**: Maintains a window of up to N unacknowledged packets.
- **Receiver**: Sends cumulative ACKs and may discard out-of-order packets.

Selective Repeat Protocol

- **Sender**: Retransmits only unacknowledged packets.
- **Receiver**: Buffers out-of-order packets for in-order delivery.

Advanced Topics

- **QUIC**: A protocol on top of UDP to improve HTTP performance, combining reliability, congestion control, and security in one handshake.

1. Multiplexing and Demultiplexing

Transport layer handles multiple data streams between applications through:

- Port numbers for identifying specific applications

- Socket pairs (source IP:port, dest IP:port) for unique connections
- Connection-oriented (TCP) vs connectionless (UDP) demultiplexing

2. User Datagram Protocol (UDP)

Characteristics of UDP include:

- Connectionless protocol with minimal overhead
- No guarantee of delivery, ordering, or integrity
- Best effort service model
- Commonly used for streaming, DNS, and SNMP

3. Transmission Control Protocol (TCP)

Key features of TCP:

- Connection-oriented with three-way handshake
- Reliable data transfer through:
- Sequence numbers and acknowledgments
- Retransmission of lost segments
- Flow control using sliding window
- Congestion control mechanisms

4. TCP Connection Management

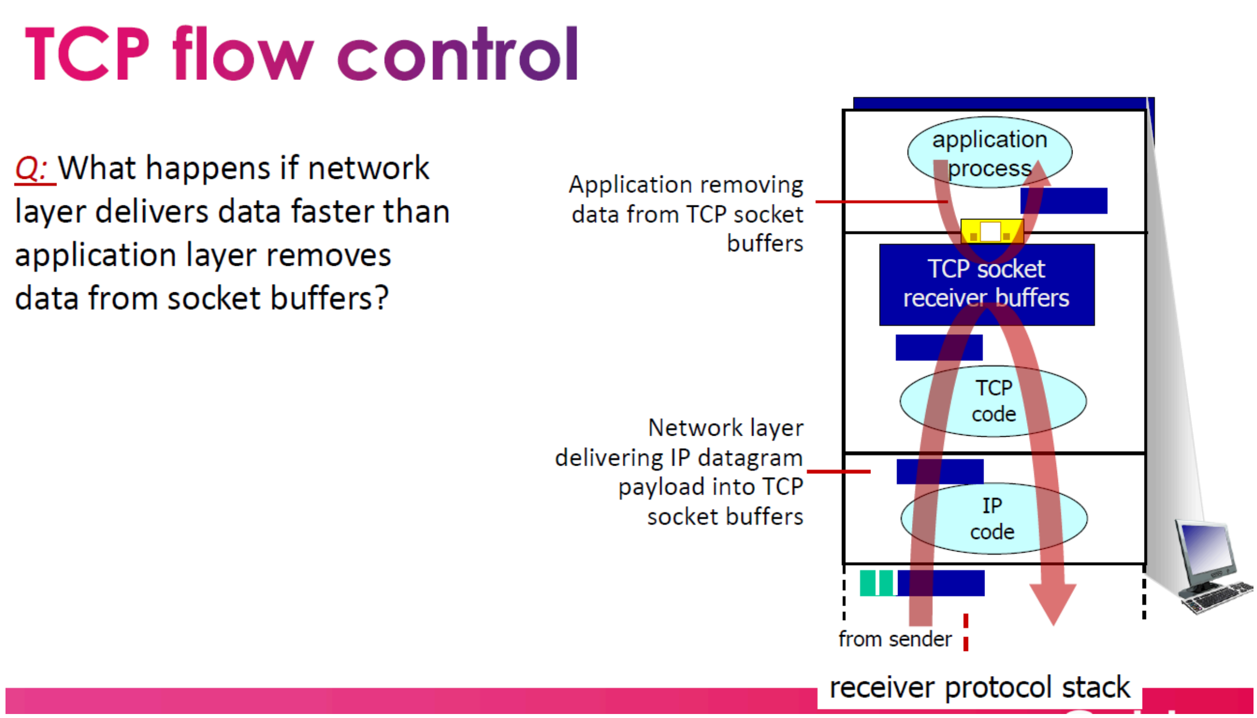
Connection lifecycle phases:

- Connection establishment (SYN, SYN+ACK, ACK)
- Data transfer phase
- Connection termination (FIN, ACK sequences)

5. Flow Control

TCP implements flow control through:

- Receiver window (rwnd) advertising
- Sliding window mechanism
- Buffer management at sender and receiver



6. Congestion Control

TCP congestion control includes:

- Slow start phase
- Congestion avoidance
- Fast retransmit and fast recovery
- AIMD (Additive Increase, Multiplicative Decrease)

7. TCP Variants

Different TCP implementations:

- TCP Tahoe
- TCP Reno

- TCP NewReno
- TCP CUBIC

8. Security Considerations

Transport layer security aspects:

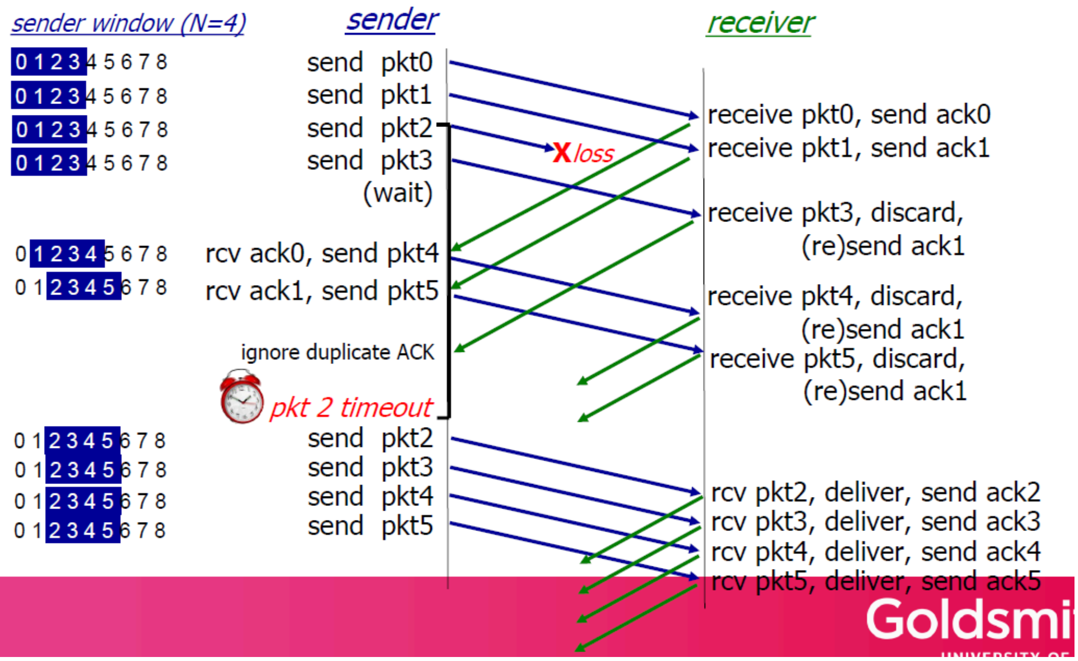
- TCP SYN flooding attacks
- Connection hijacking
- UDP amplification attacks
- Implementation of TLS/SSL

Summary

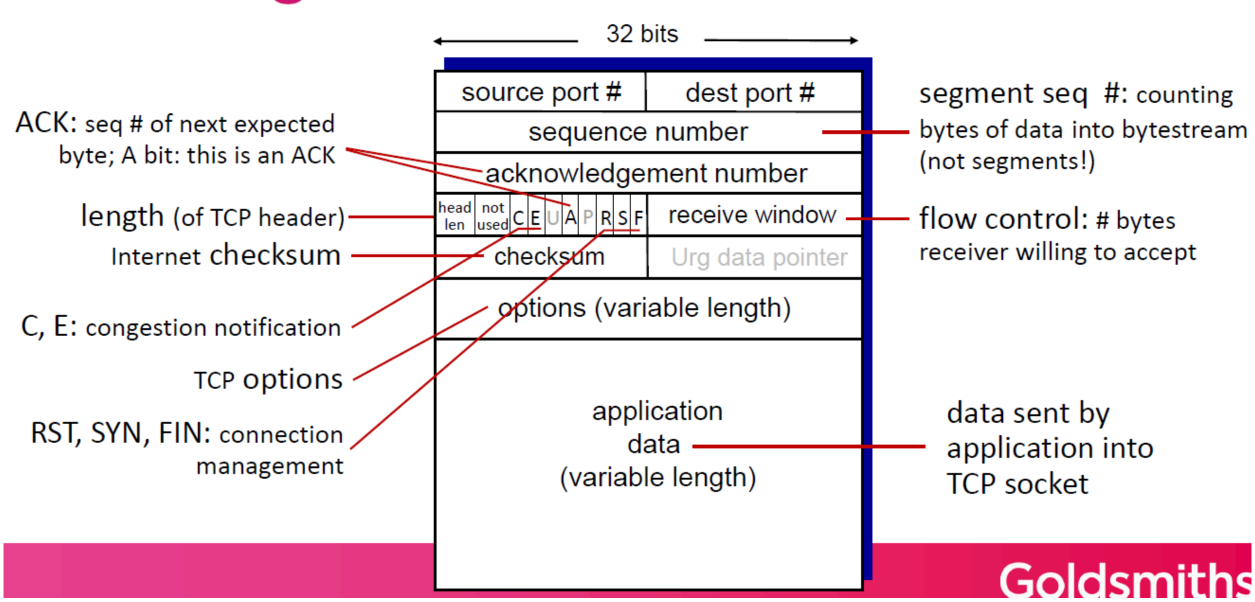
- **Transport Layer Services**: Multiplexing, demultiplexing, reliable data transfer, flow control, and congestion control.
- **Protocols**: UDP (User Datagram Protocol) and TCP.
- **Future Topics**: Network-layer data plane and control plane.

This lecture covered fundamental concepts of transport layer protocols, focusing on TCP and UDP, their mechanisms for reliable data transfer, flow control, and congestion management. Understanding these concepts is crucial for network application development and troubleshooting.

Go-Back-N in action



TCP segment structure



TCP sequence numbers, ACKs

Sequence numbers:

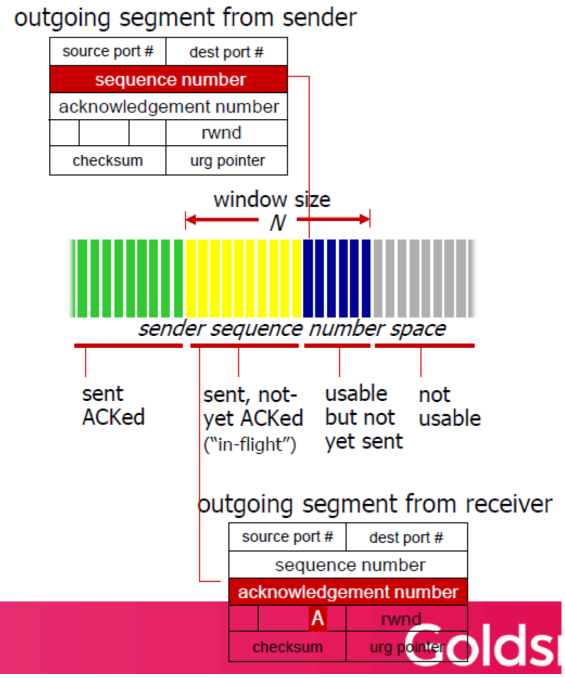
- byte stream "number" of first byte in segment's data

Acknowledgements:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

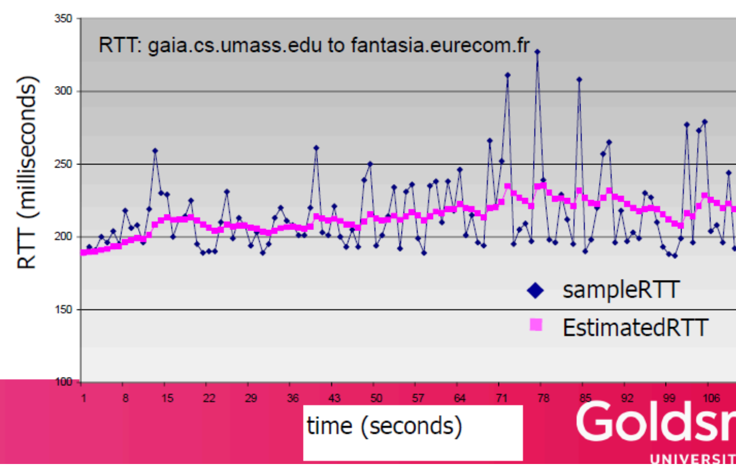
- *A:* TCP spec doesn't say, - up to implementor



TCP round trip time, timeout

$$\text{EstimatedRTT} = (1-\alpha)\text{EstimatedRTT} + \alpha\text{SampleRTT}$$

- exponential weighted moving average (EWMA)
- influence of past sample decreases exponentially fast
- typical value: α = 0.125



TCP round trip time, timeout

- timeout interval: **`EstimatedRTT`** plus “safety margin”
  - large variation in **`EstimatedRTT`**: want a larger safety margin

```
TimeoutInterval = EstimatedRTT + 4DevRTT
```

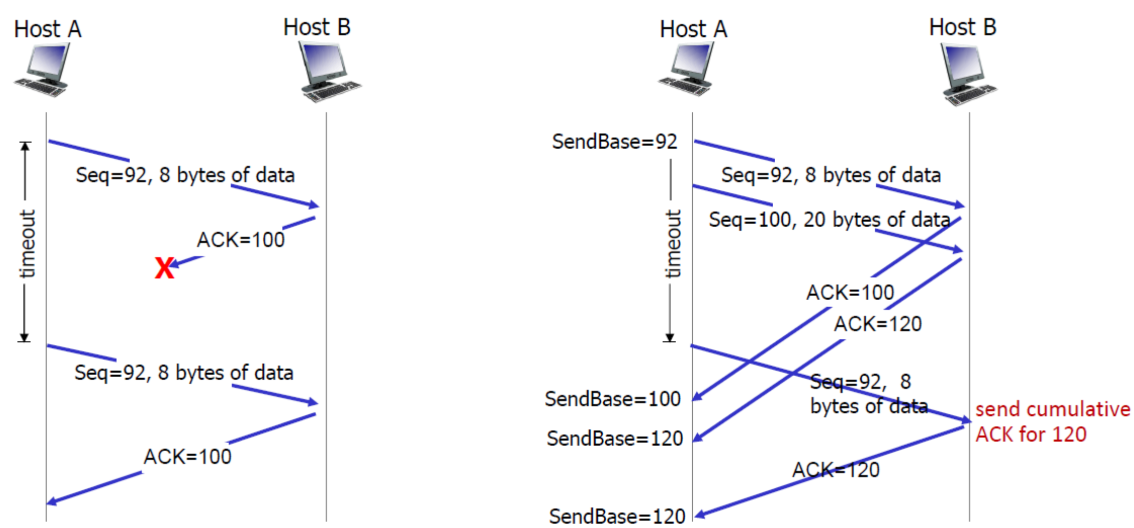
estimated RTT “safety margin”

- **`DevRTT`**: EWMA of **`SampleRTT`** deviation from **`EstimatedRTT`**:

```
DevRTT = (1-β)DevRTT + β|SampleRTT-EstimatedRTT|
```

(typically, β = 0.25)

TCP: retransmission scenarios



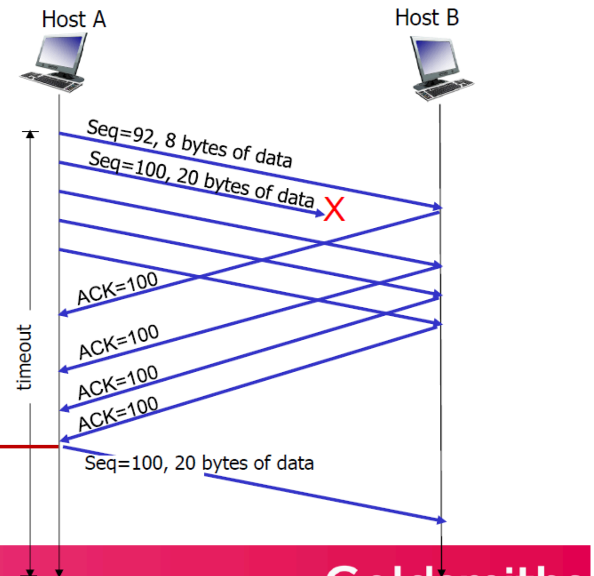
lost ACK scenario premature timeout

TCP fast retransmit

TCP fast retransmit

if sender receives 3 additional ACKs for same data ("triple duplicate ACKs"), resend unACKed segment with smallest seq #

- likely that unACKed segment lost, so don't wait for timeout



Receipt of three duplicate ACKs indicates 3 segments received after a missing segment – lost segment is likely. So retransmit!

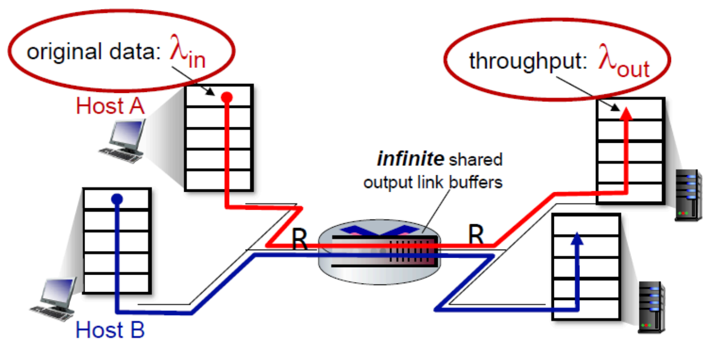
Goldsmiths

Transport Layer: 3-306

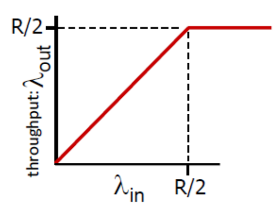
Causes/costs of congestion: scenario 1

Simplest scenario:

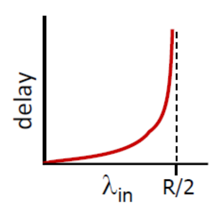
- one router, infinite buffers
- input, output link capacity: R
- two flows
- no retransmissions needed



Q: What happens as arrival rate λ_{in} approaches R/2?



maximum per-connection throughput: R/2



large delays as arrival rate λ_{in} approaches capacity