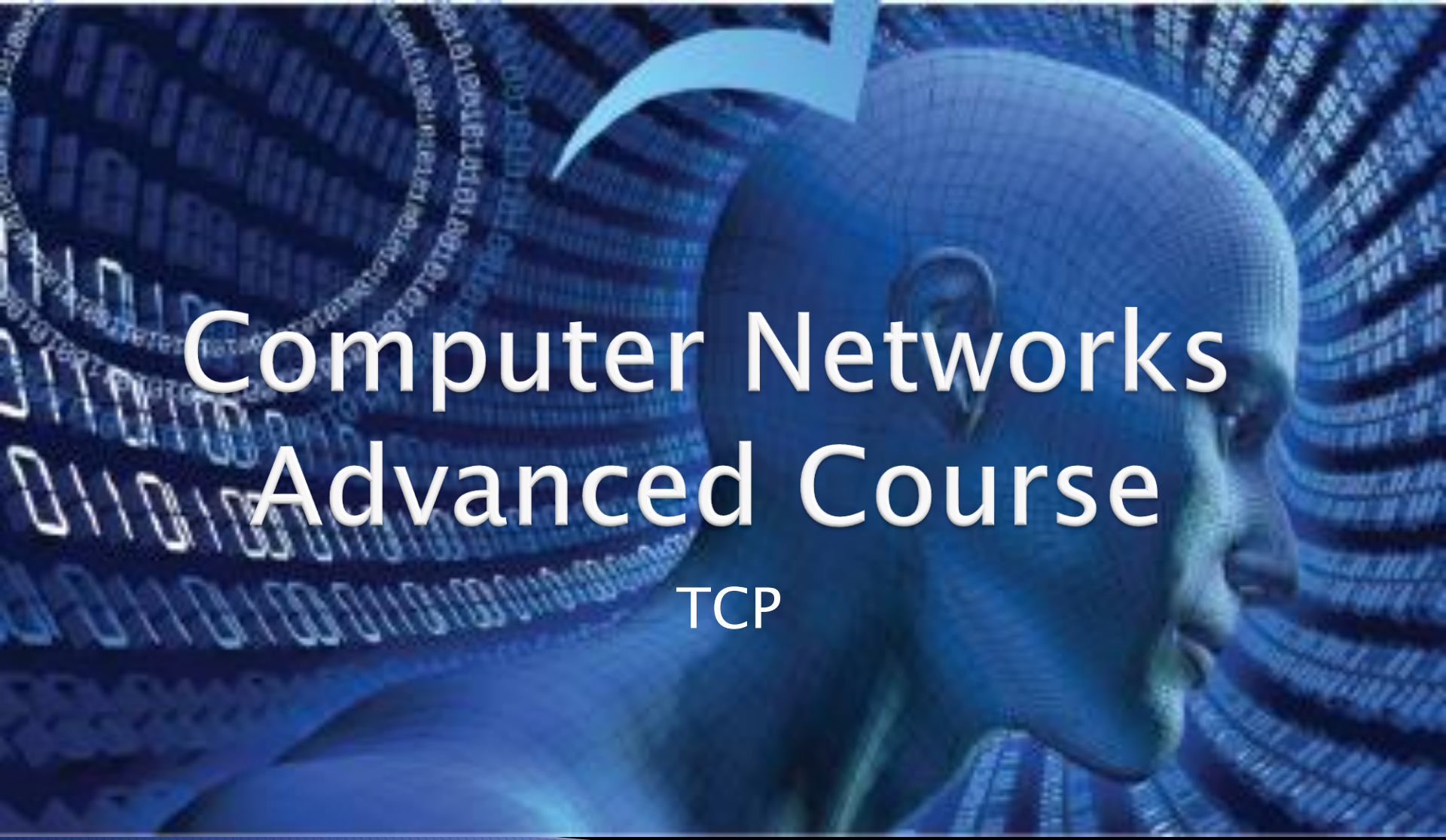


# Computer Networks Advanced Course

## TCP

Barak Gonen
Based on "Computer Networks"
Rosenboim, Gonen, Hod

# Presentation Goals

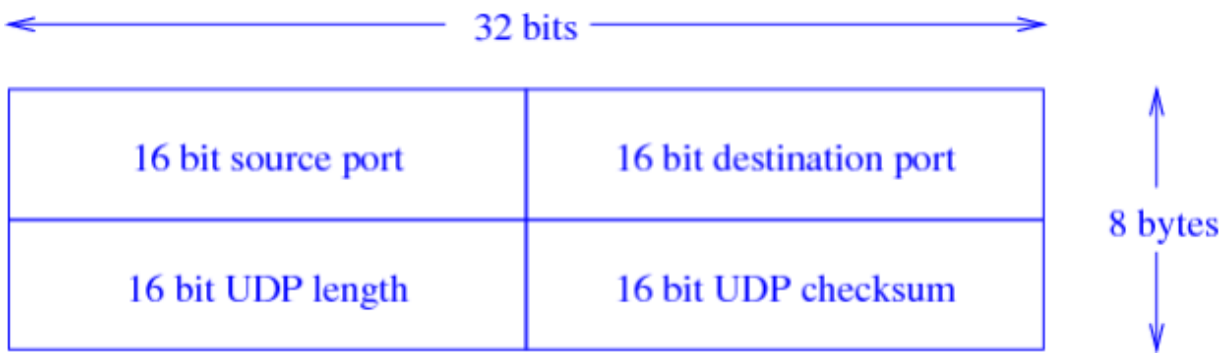
- How reliable communication is ensured
  - TCP Seq numbers
  - TCP ACK
  - TCP 3 way handshake
- Hands on- Wireshark, SYN flood attack

# Brief Recap

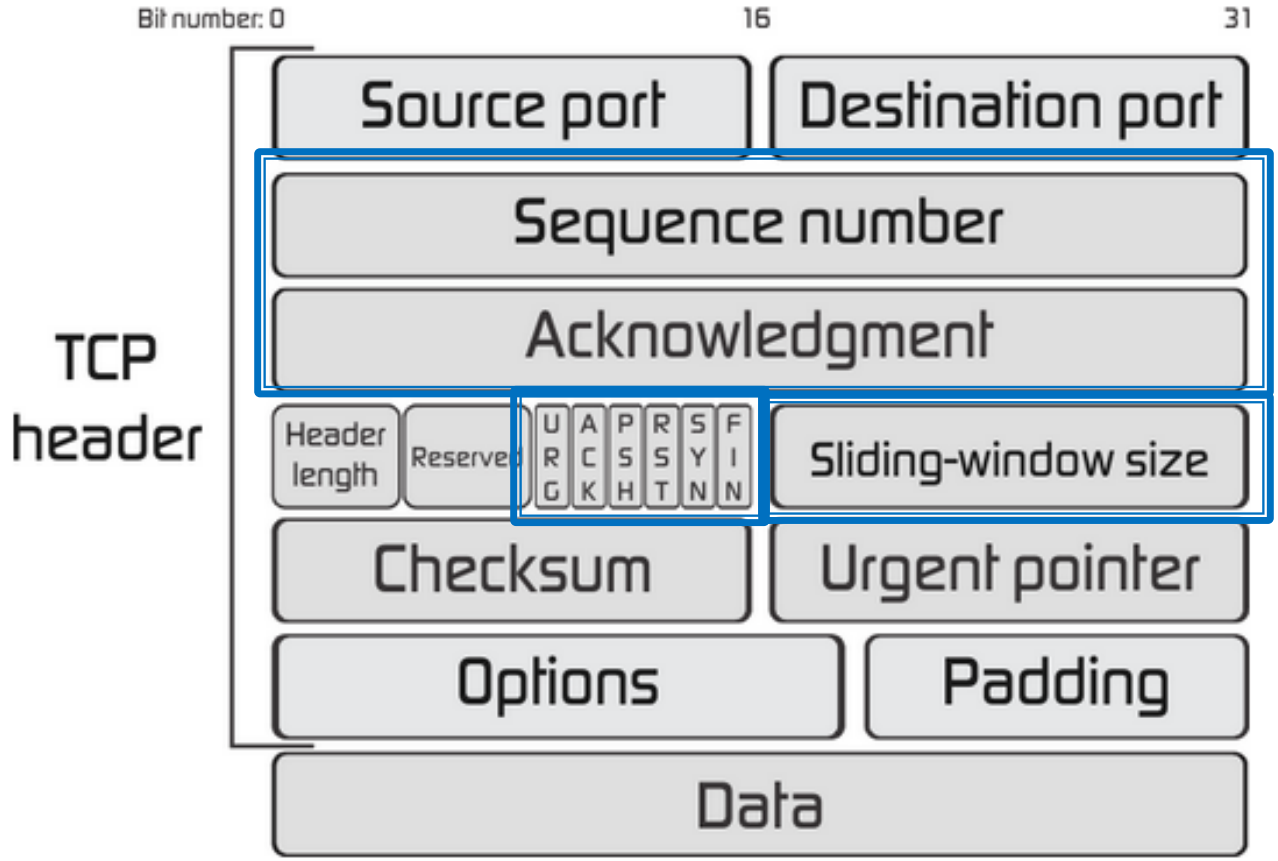
- Transport layer may, optionally, provide reliable service
  - TCP – Transmission Control Protocol – reliable
  - UDP – User Datagram Protocol – Best effort
- Reliable service:
  - All packets arrived
  - In order
  - No errors

# UDP Header

- Ports – src, dst
- Length – header + application
- Checksum – not error correction
- Hands on Wireshark

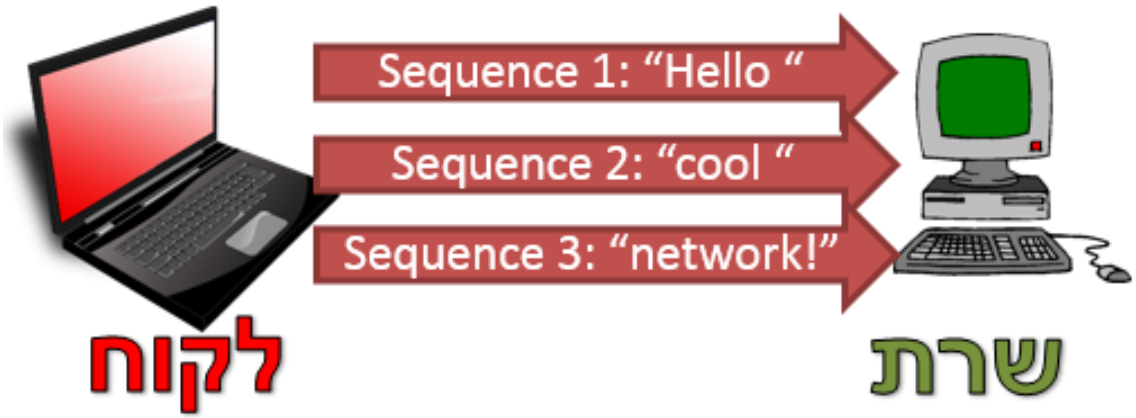


# TCP Header


Bit number: 0
16
31
TCP header
Source port
Destination port
Sequence number
Acknowledgment
Header length
Reserved
URG
ACK
PSH
RST
SYN
FIN
Sliding-window size
Checksum
Urgent pointer
Options
Padding
Data

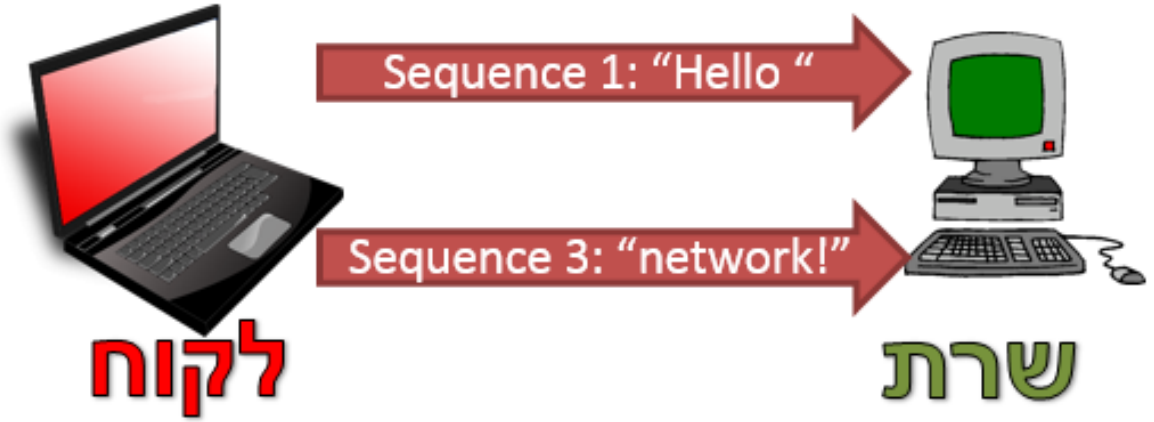
# Sequence Numbers Idea

- This is not how it works, but demonstrates the basic idea



# Sequence Numbers Idea

- The receiver may know if a sequence is missing
- What if sequence 2 was the last?

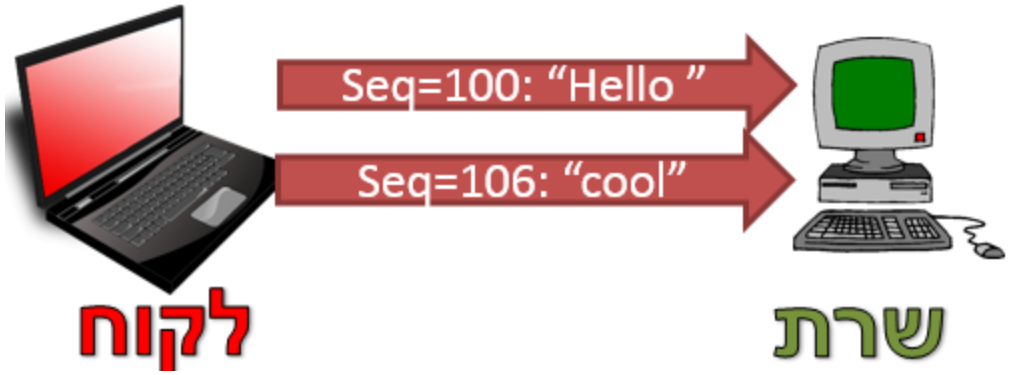


# ACK

- ACK signals that no need to retransmit

# TCP Sequential Numbers

- TCP, in practice
- Every byte has a sequence number
- SEQ field has the value of the first byte




Note – there is space after “Hello”

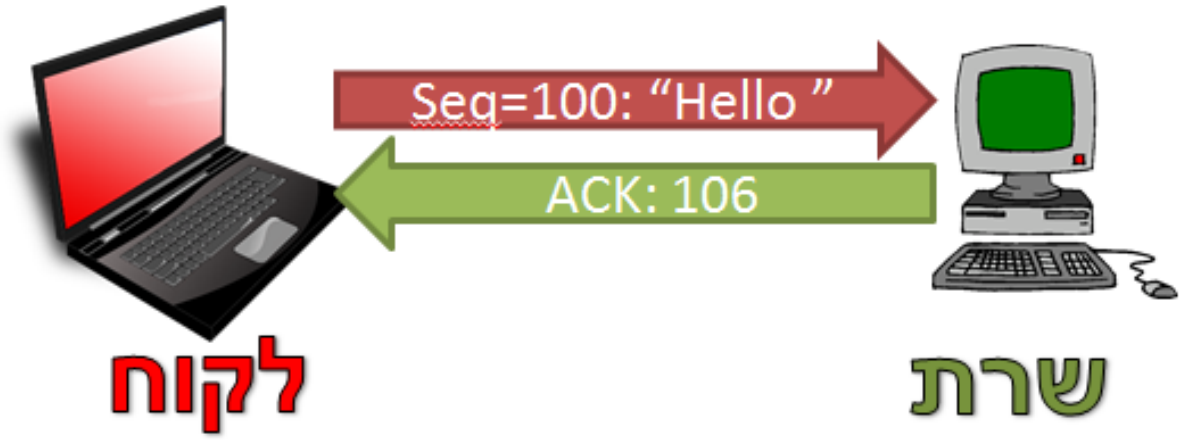
- What is the next SEQ?
  - (106+4) 110

# TCP Sequence Numbers

- Ex 6.14
- Use Wireshark to watch seq numbers
- Make sure: next SEQ = current SEQ + length

# TCP ACK

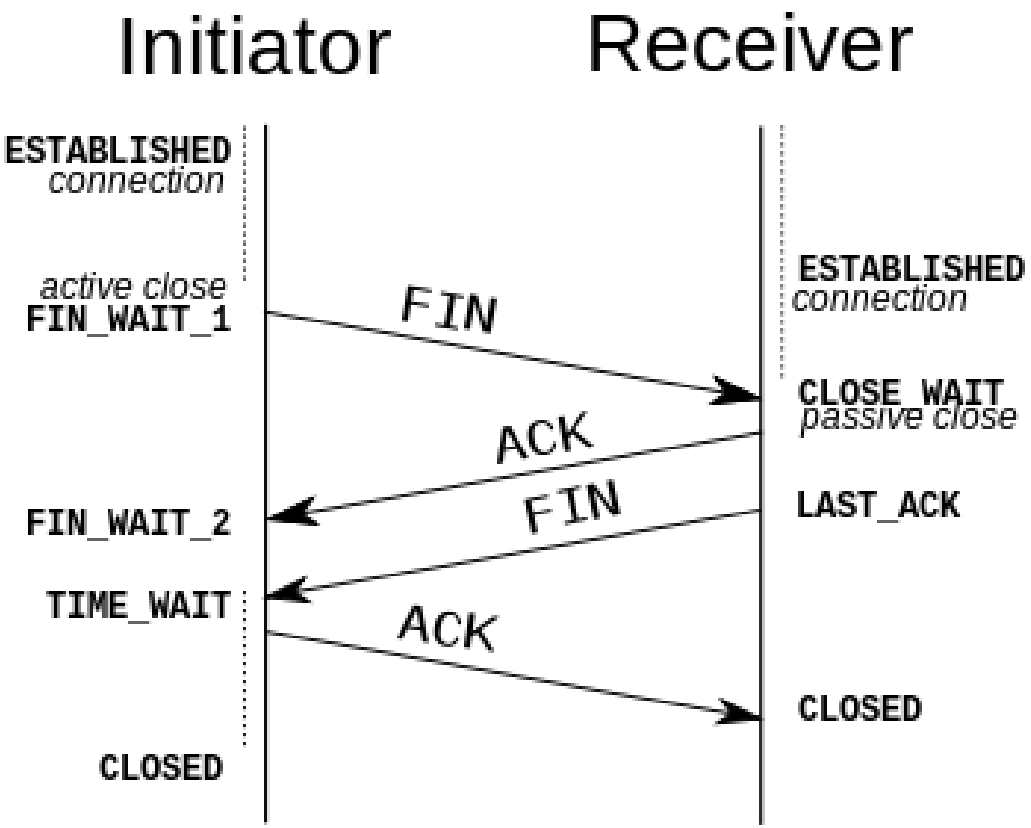
- ACK relates to bytes
  - ACK 106 – “Got up to byte 105, including. Expecting 106 in the next packet”



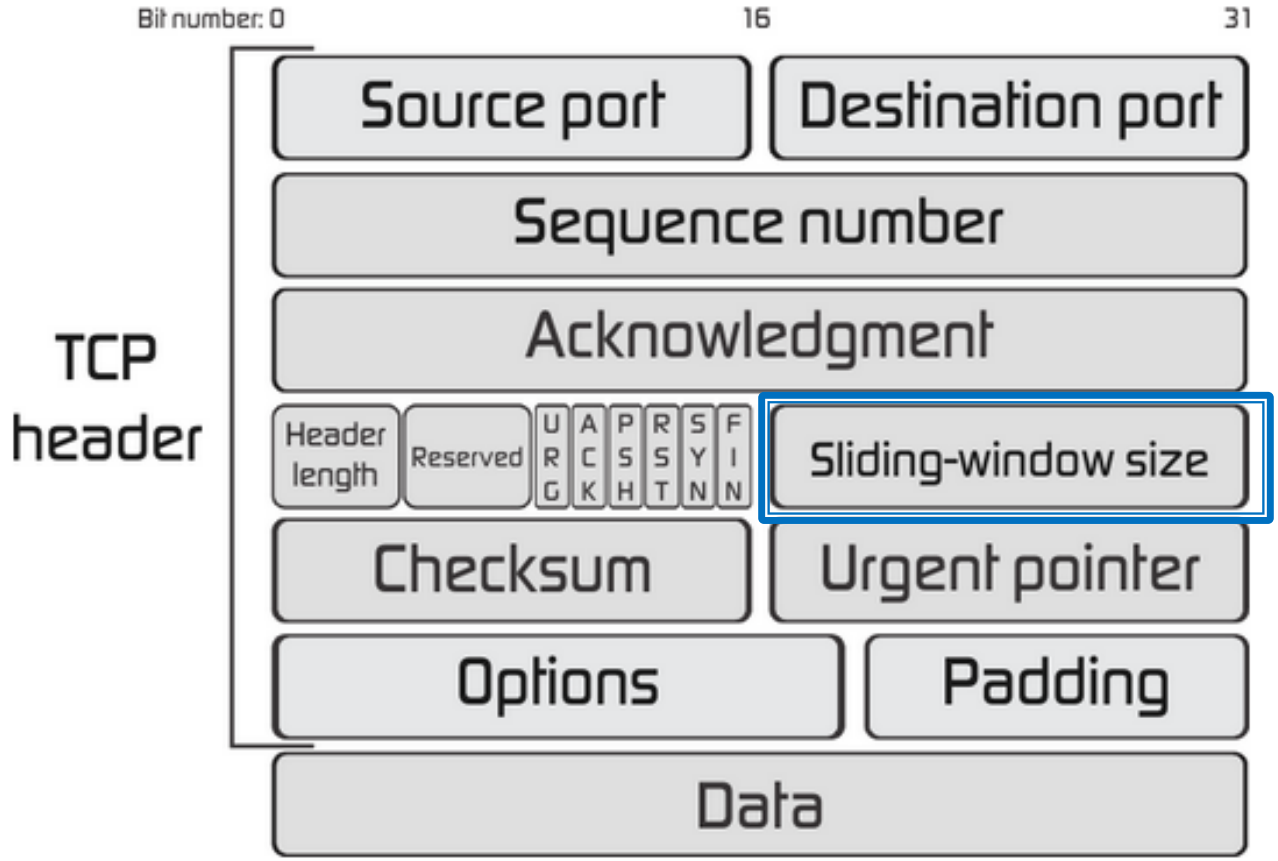
# TCP ACK

- Ex 6.15
- Make sure: the ACKs match the SEQ + length

# TCP Closing Connection

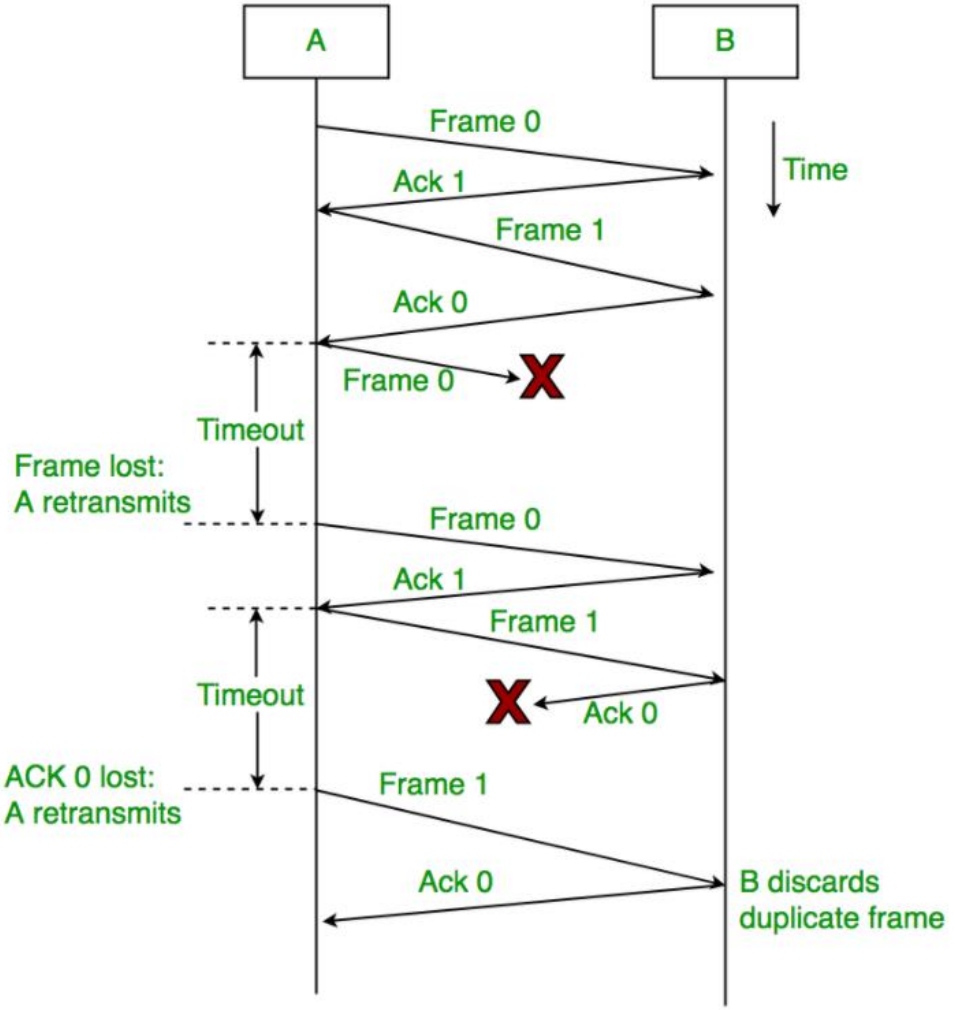



# TCP Header


Bit number: 0
16
31
TCP header
Source port
Destination port
Sequence number
Acknowledgment
Header length
Reserved
URG
ACK
PSH
RST
SYN
FIN
Sliding-window size
Checksum
Urgent pointer
Options
Padding
Data

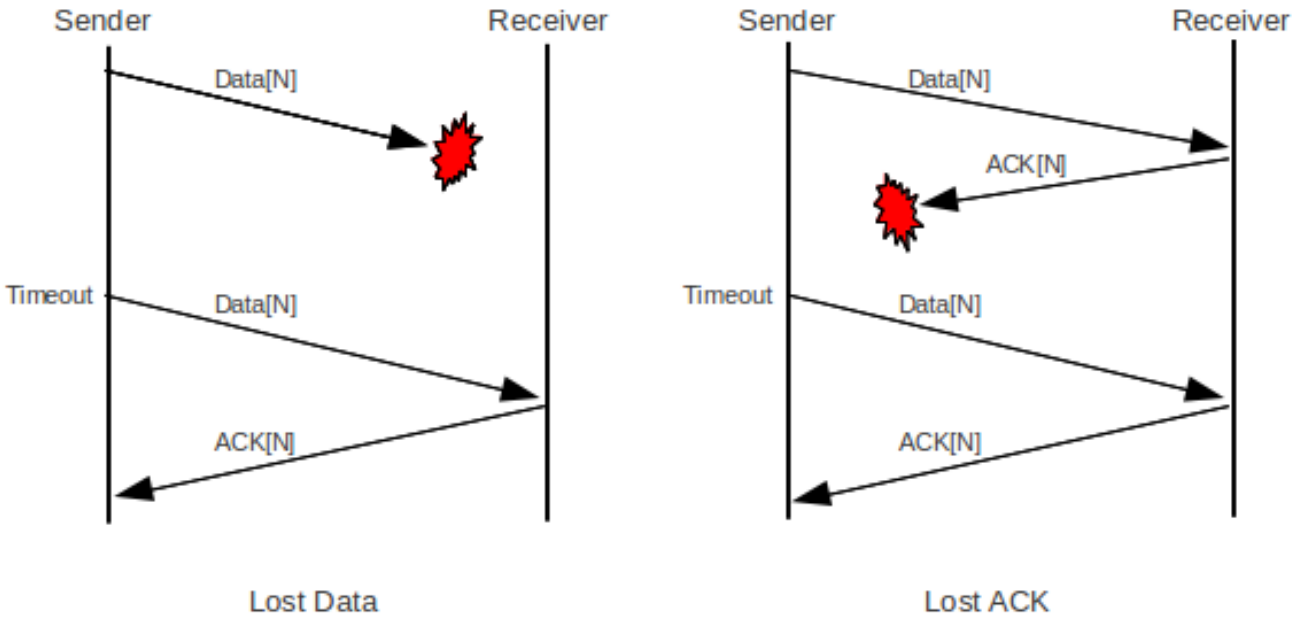
# Stop and Wait

- What is the drawback of the algorithm?



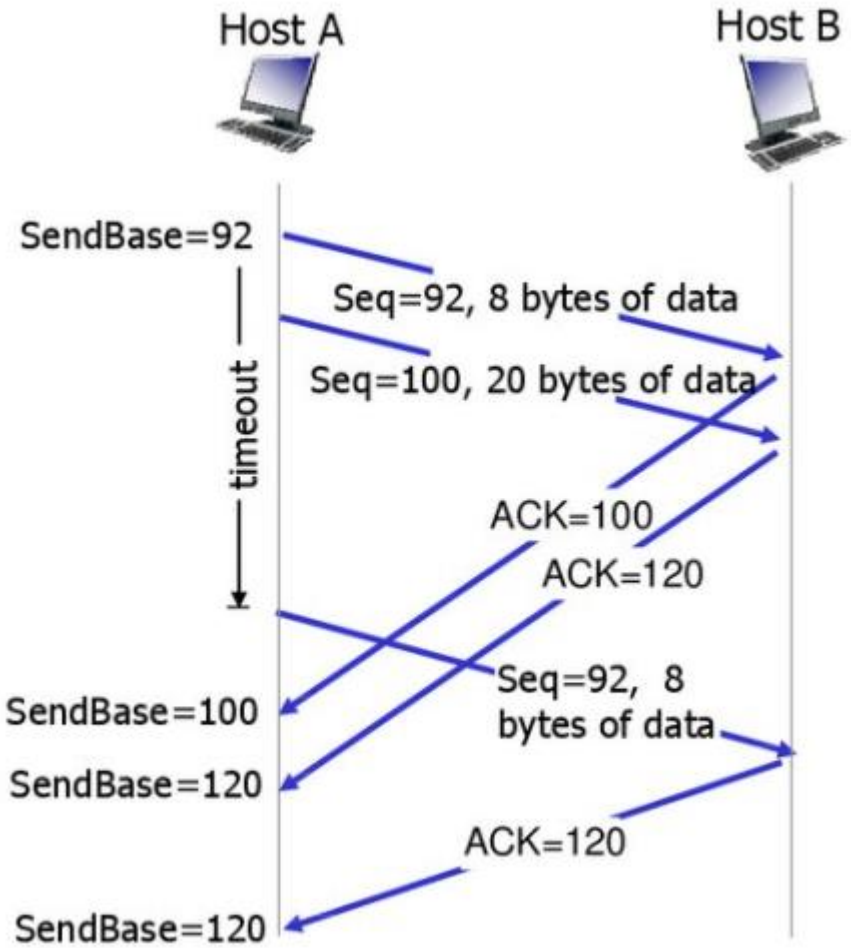
# Retransmit

- Will occur if:



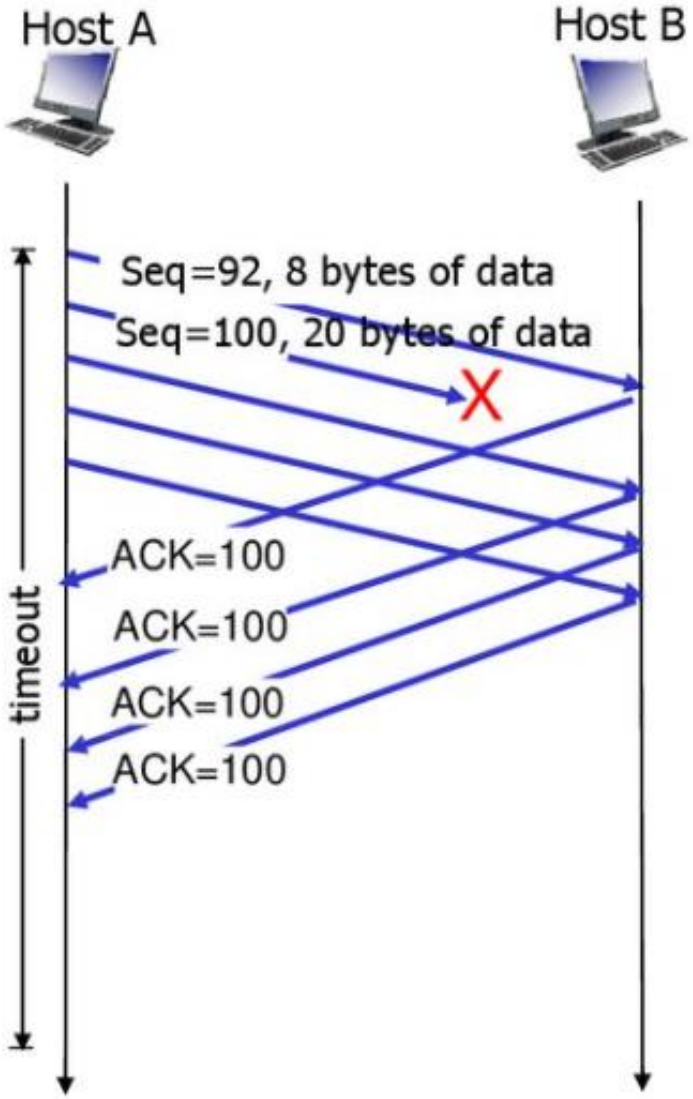

- Think of another scenario

# Premature Timeout



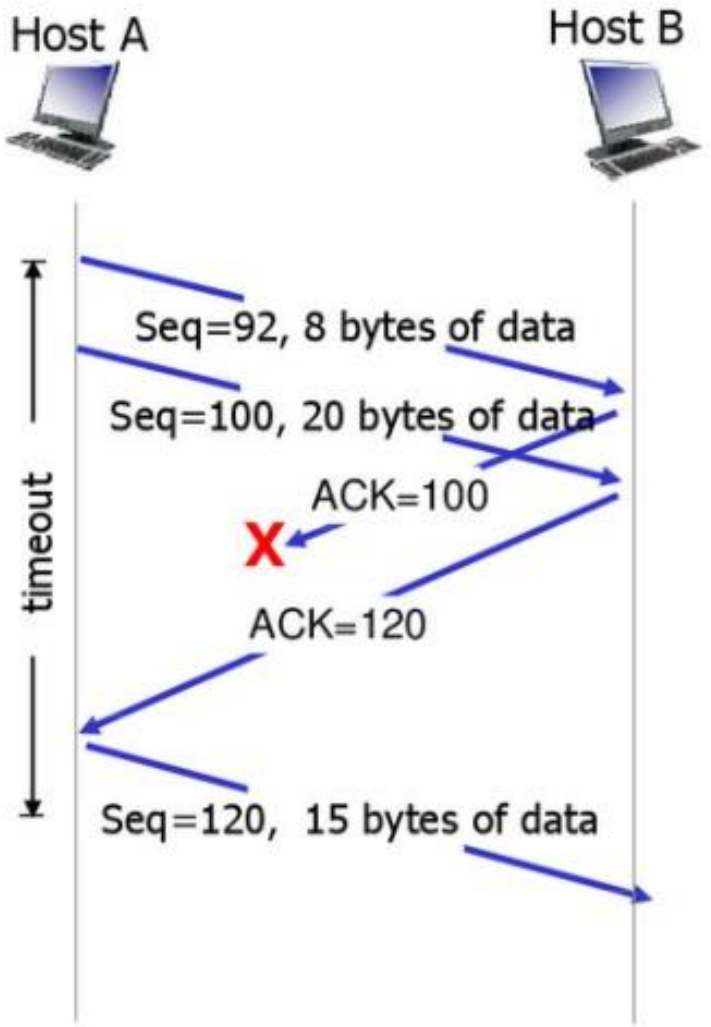
# Fast retransmit

- Should A wait until timeout with the lost packet?



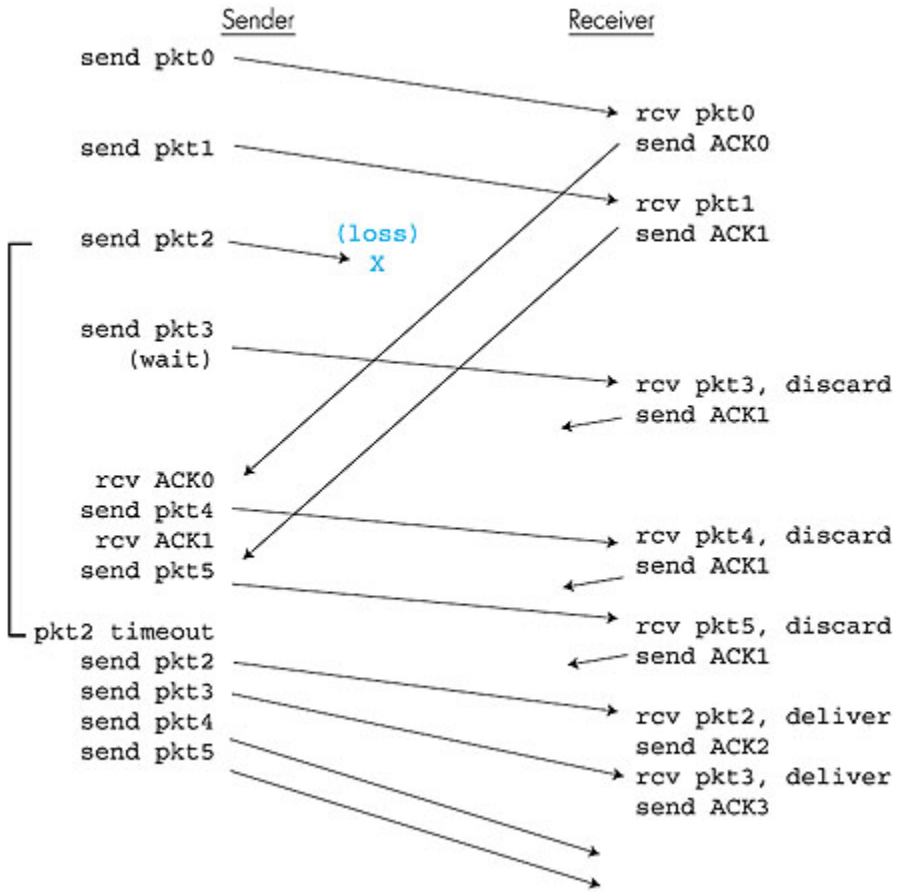
# Cumulative ACK

- Why didn't A resend packet of SEQ 92?



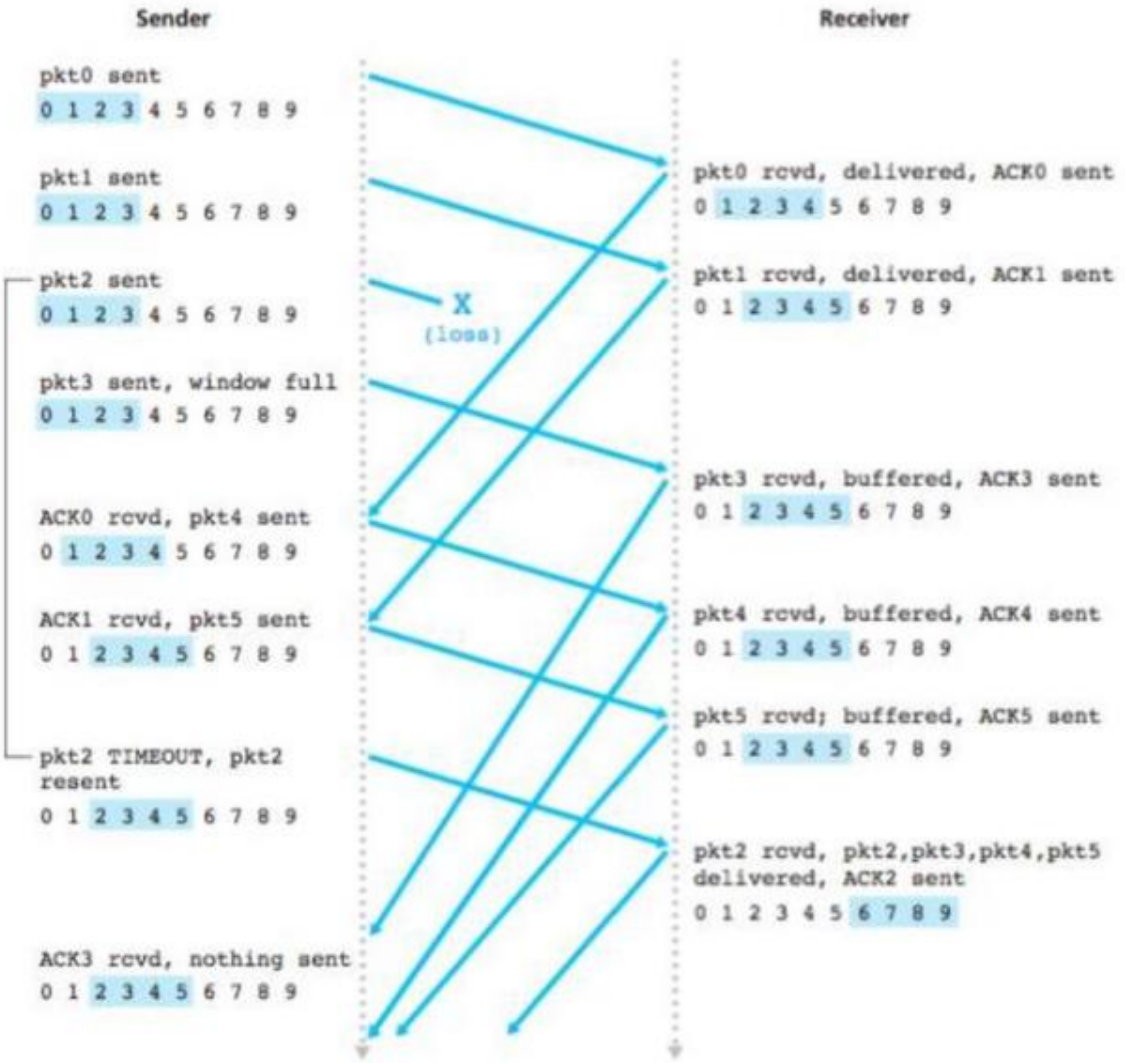
# Go-Back-N

- Watch Go-Back-N

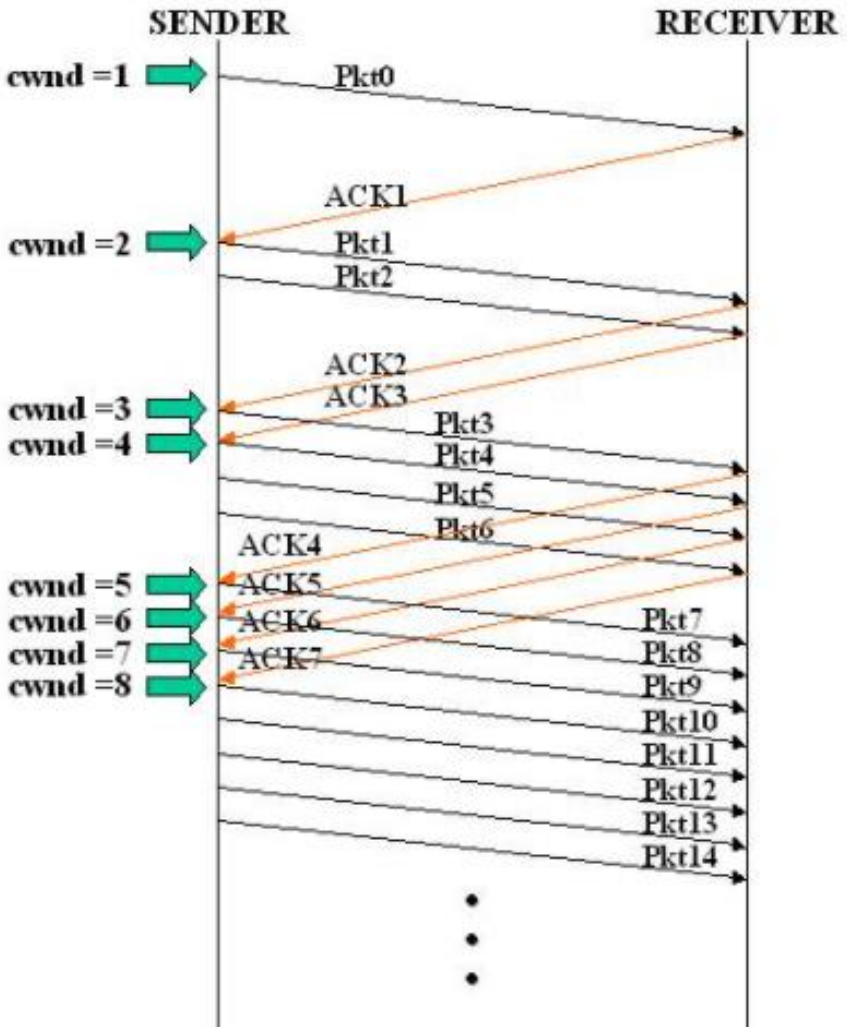


# Selective Repeat

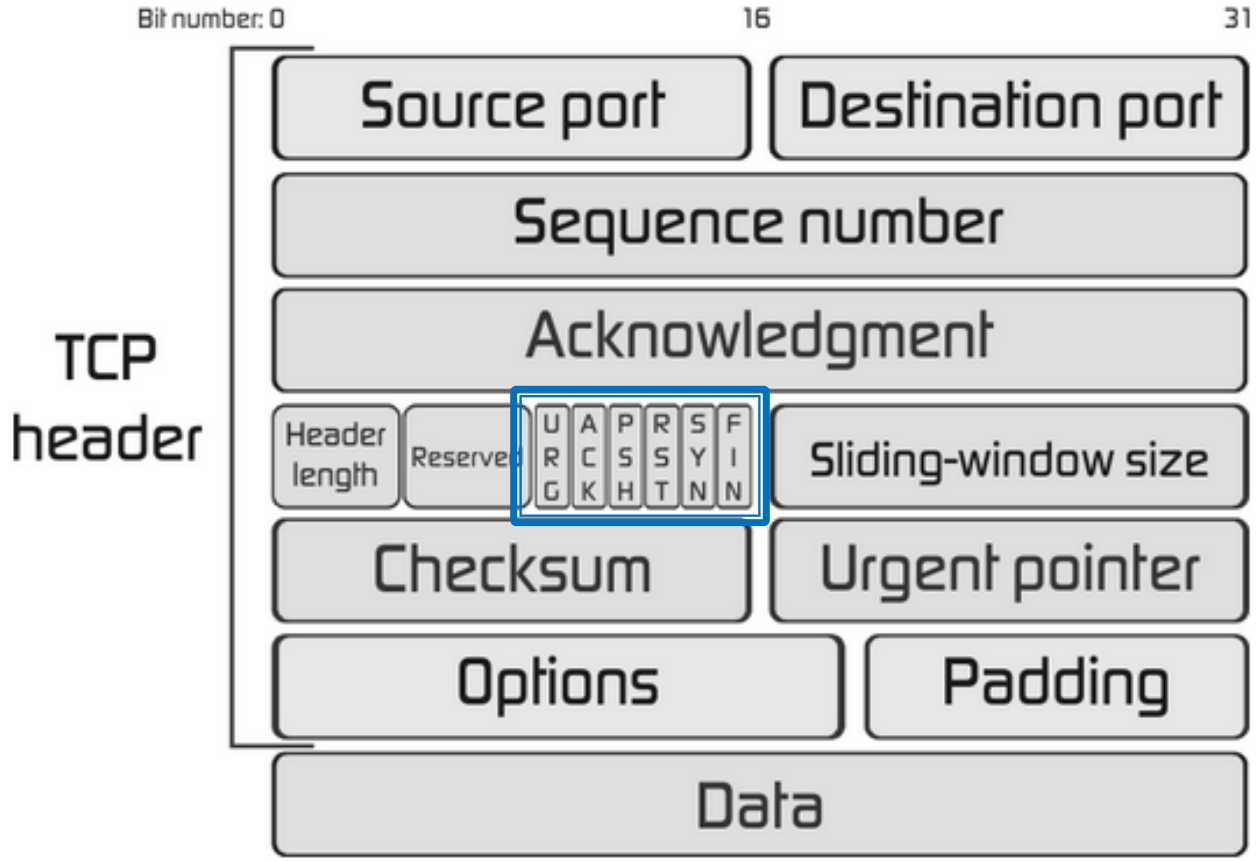
- Isn't selective repeat always better than cumulative ACK?



# Congestion Control – Slow Start

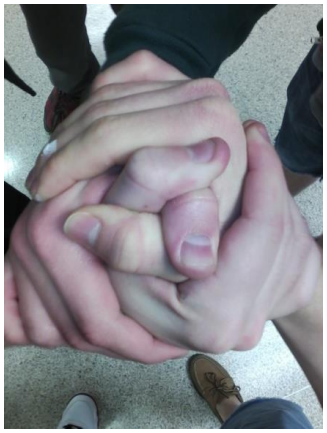


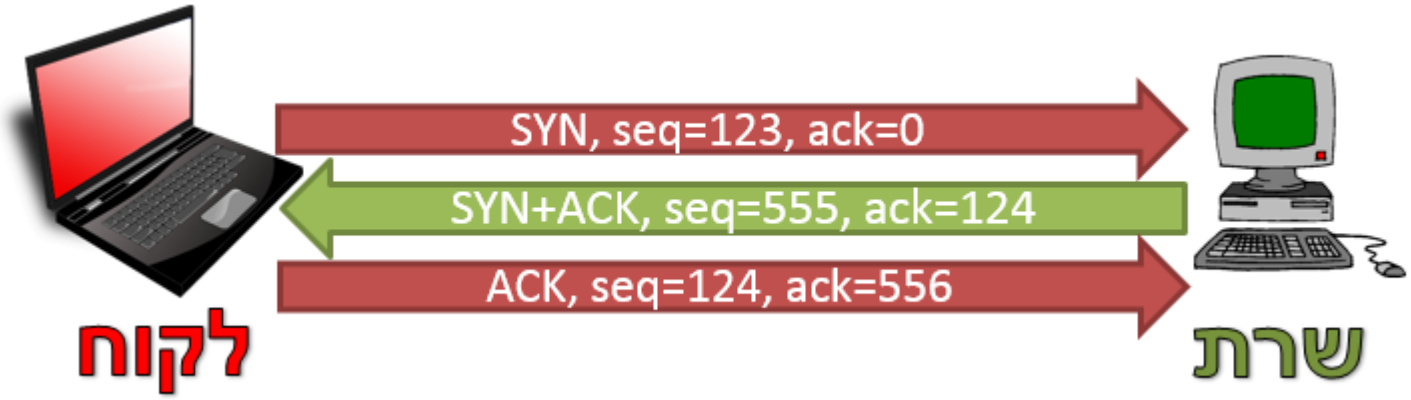
# TCP Header


Bit number: 0
16
31
TCP header
Source port
Destination port
Sequence number
Acknowledgment
Header length
Reserved
URG
ACK
PSH
RST
SYN
FIN
Sliding-window size
Checksum
Urgent pointer
Options
Padding
Data

# TCP: Three Way Handshake

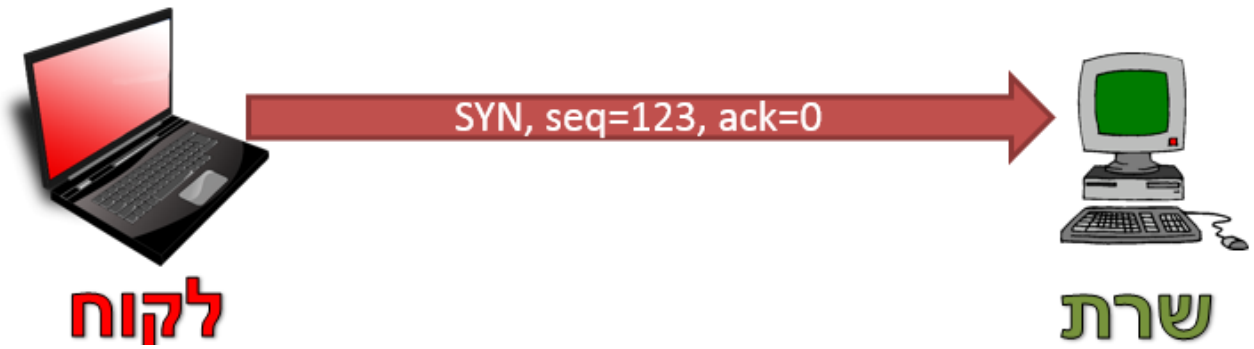
- Establishing a connection





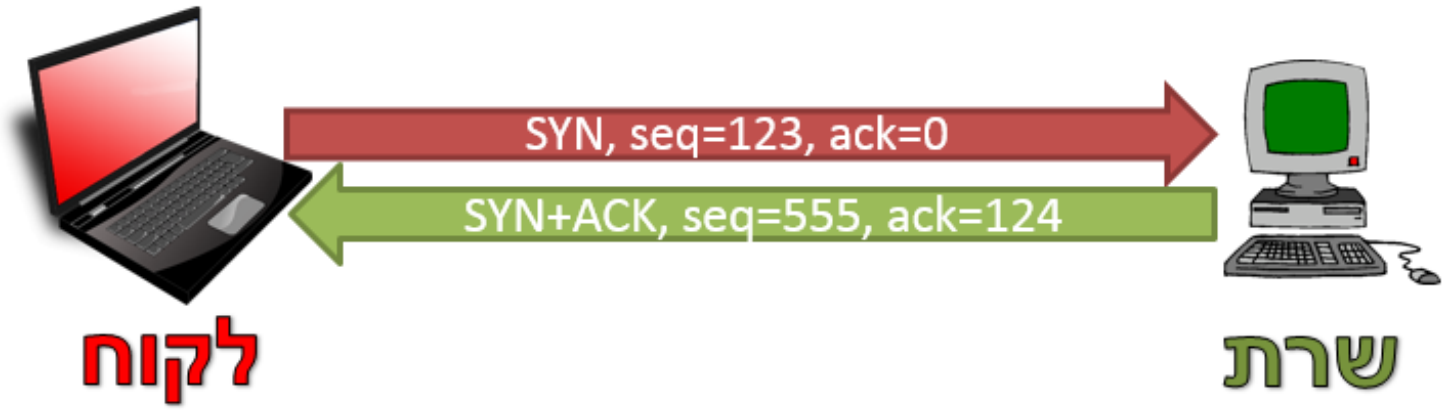
# SYN Packet

- “I want to connect”
- SYN flag == 1
- No data in packet, but still considered as having length of 1
- Random initial SEQ
- ACK is always 0



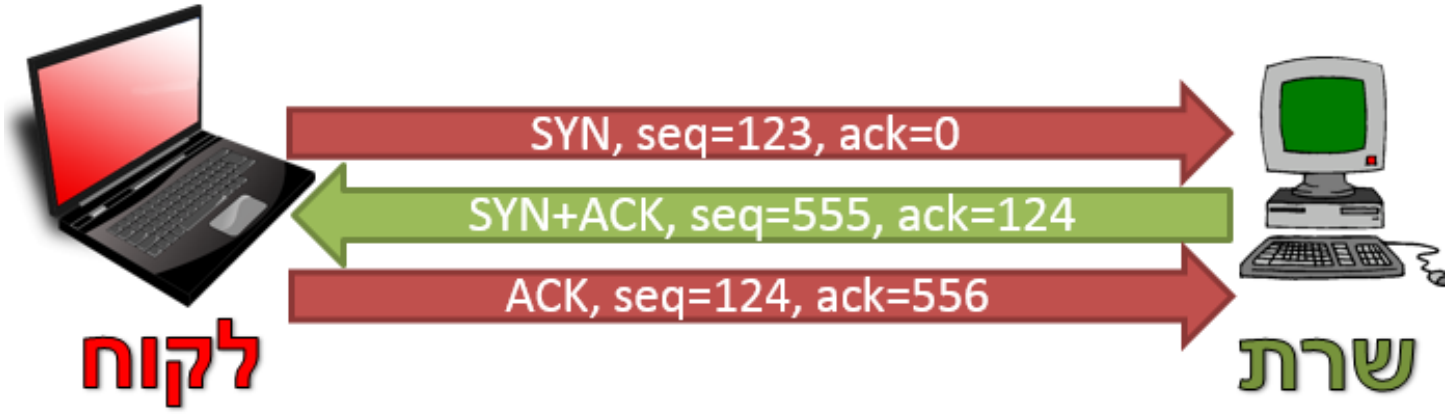
# SYN ACK

- “I agree to connect”
- SYN flag, ACK flag
- Packet length == 1
- Random SEQ (not related to SYN packet’s SEQ)
- ACK value is the SEQ of the SYN packet + 1
  - Recall SYN length is considered 1



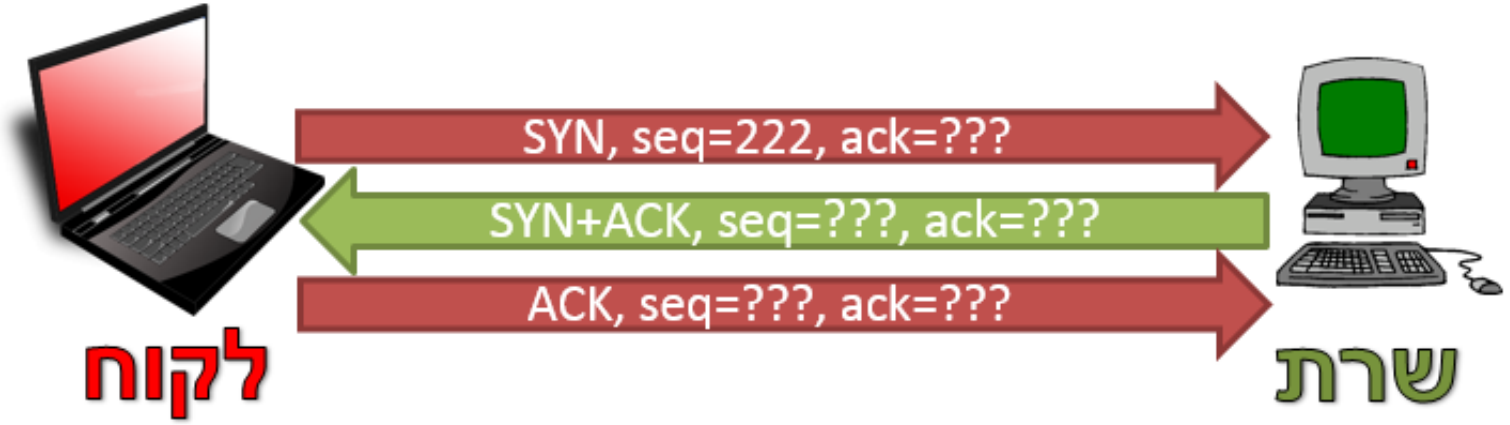
# ACK

- “Got your SYN-ACK, let’s initiate communication”
- ACK flag = 1, SYN = 0
- SEQ is the last byte that was sent
- ACK is the SEQ of the SYN-ACK plus 1



# Summary – ACK, SEQ Calculation

- Figure the ACK and SEQ values of the following
  - Use random values where proper



# Three Way Handshake

- Ex. 6.16
- Watch SEQ, ACK values and verify your expectations

# Three Way Handshake

- Ex. 6.18
- Perform 3 way handshake using Scapy
  - You may encounter a “Reset” from the OS

| Info | Length | Protocol | Destination | Source | Time |
|---|---|---|---|---|---|
| Seq=0 Win=8192 Len=0 [SYN] 80 → 55555 | 54 | TCP | 142.250.186.132 | 192.168.1.221 | 2.189167 489 |
| Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 [SYN, ACK] 55555 → 80 | 60 | TCP | 192.168.1.221 | 142.250.186.132 | 2.258783 503 |
| Seq=1 Ack=1 Win=8192 Len=0 [ACK] 80 → 55555 | 54 | TCP | 142.250.186.132 | 192.168.1.221 | 2.313947 511 |

# SYN – Flood

- SYN packets cause the server to allocate resources for a new socket
- Attacker might exploit that for a DoS attack
  - Flood with SYN packets, no ACK

- https://data.cyber.org.il/networks/SYN-Flood.pdf