Transport Layer Protocols

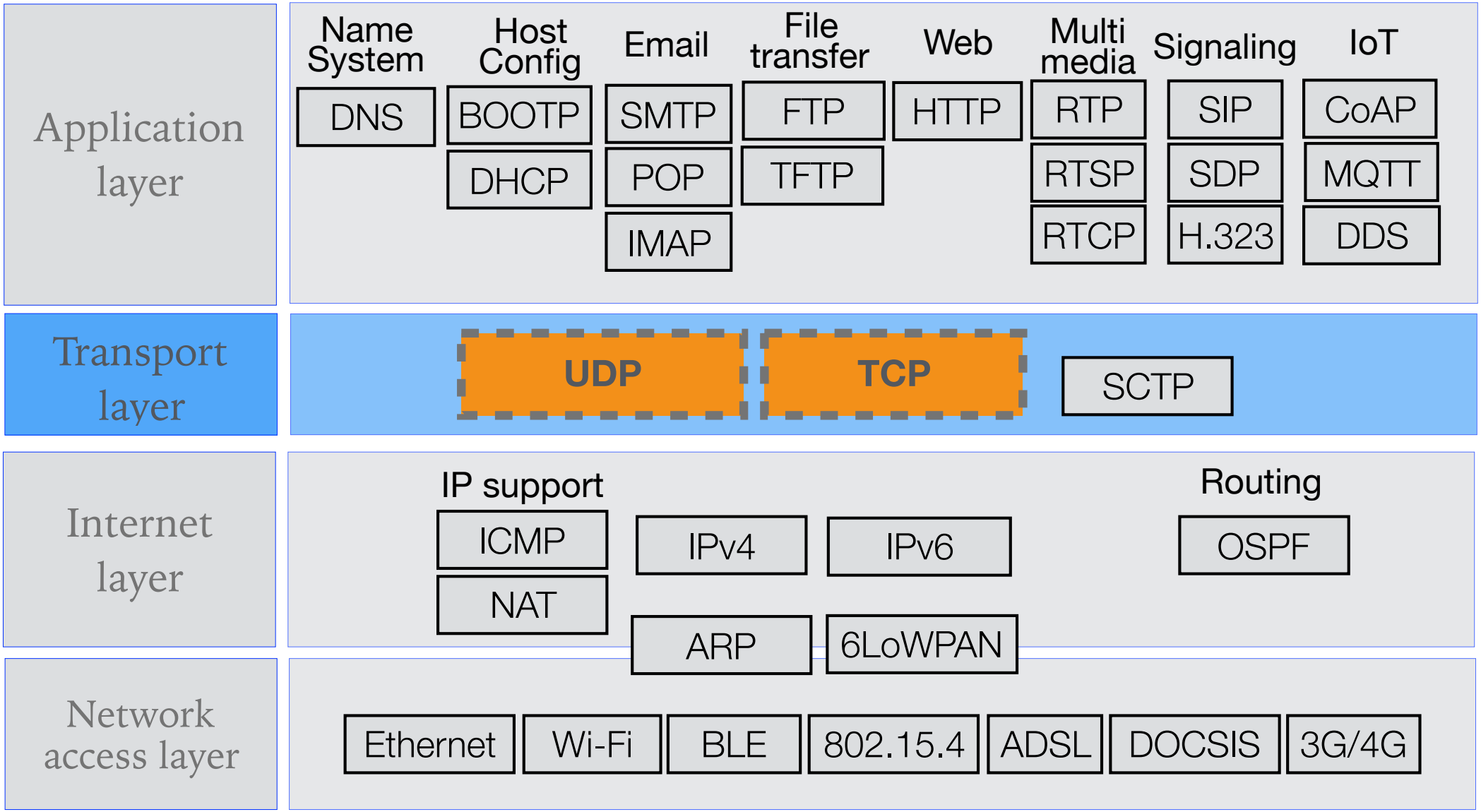
Peerapon S.

CPE 314: Computer Networks (2/61)

Topics

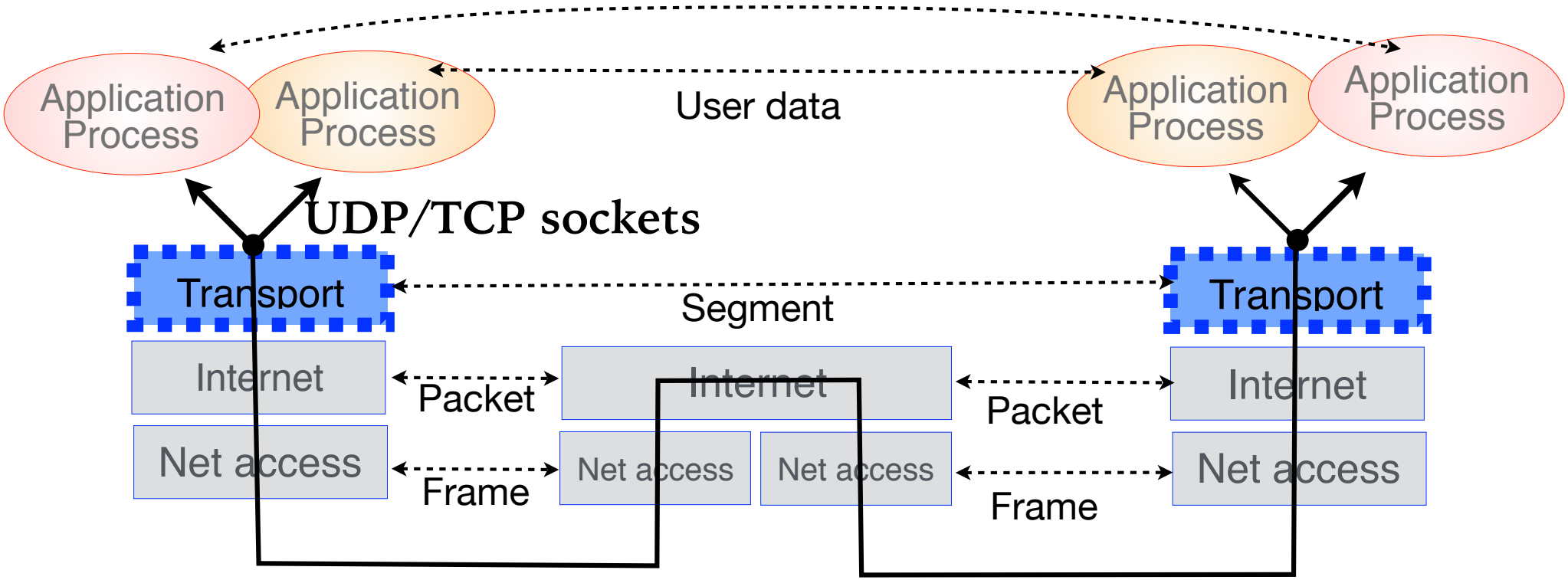
- User Datagram Protocol (UDP)
- Transport Control Protocol (TCP)
  - TCP connection management
  - TCP flow control

Protocols in TCP/IP Suite



Transport Layer (TL) Services

- Process-to-process message delivery service

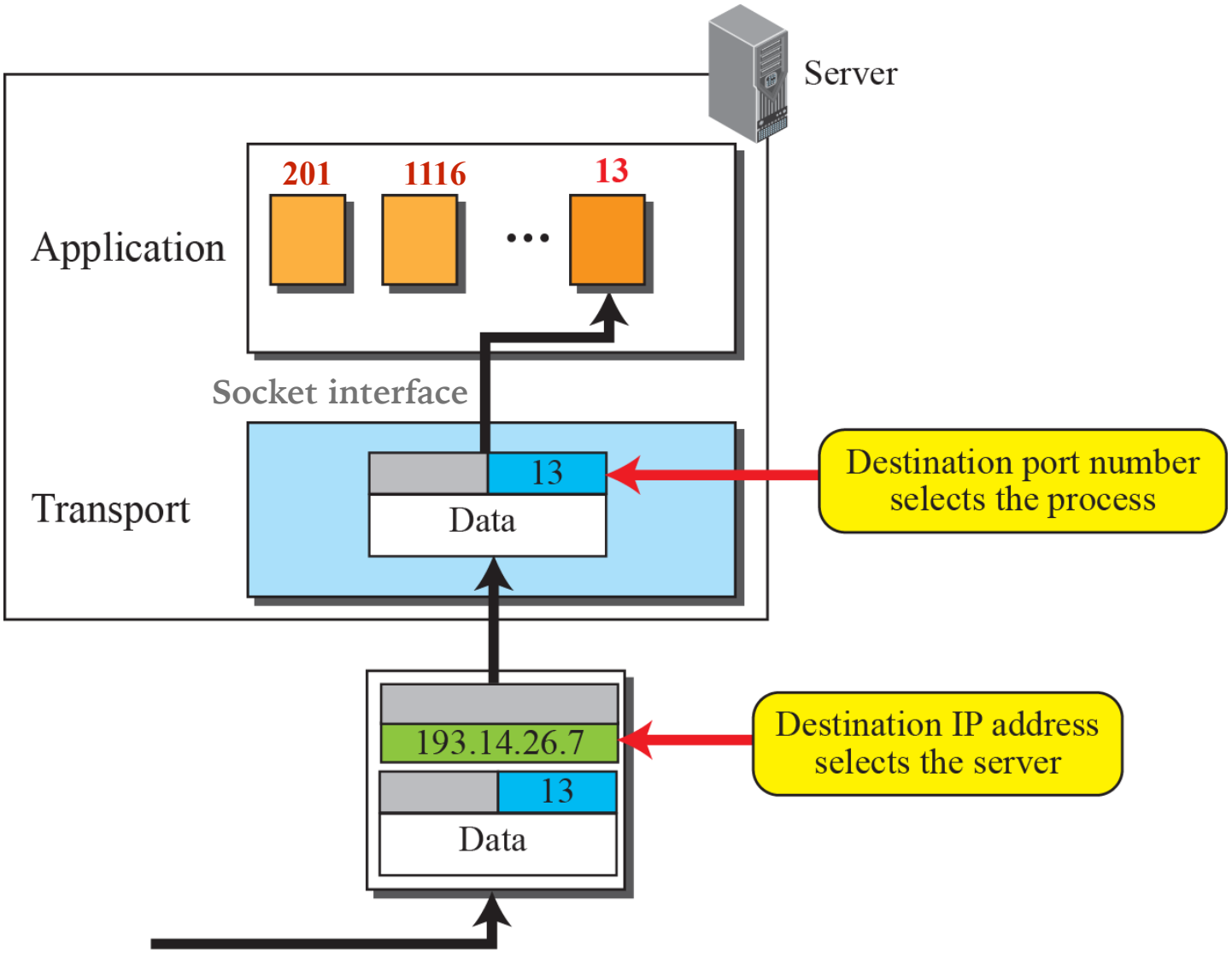


Two Main Internet Transport Layer Protocols

- **UDP: User Datagram Protocol**
  - Connection-less, Unreliable service
  - Independent messages delivery
- **TCP: Transport Control Protocol**
  - Connection-oriented, Reliable service
  - Byte-stream delivery

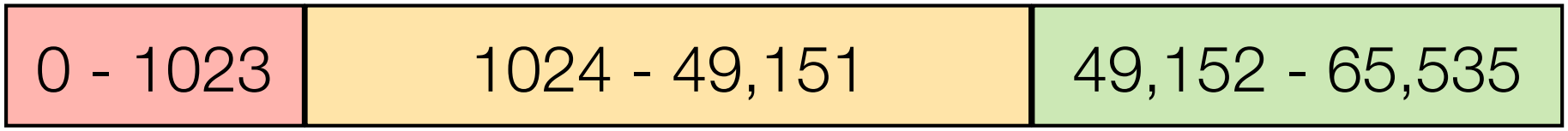
	Loss-free	Order	No duplicate	Throughput	Bound delay
UDP					
TCP	◉	◉	◉		

Basic Function: Process Multiplexing



Port Range for Server and Client

- Only some port numbers can be used by your custom app.

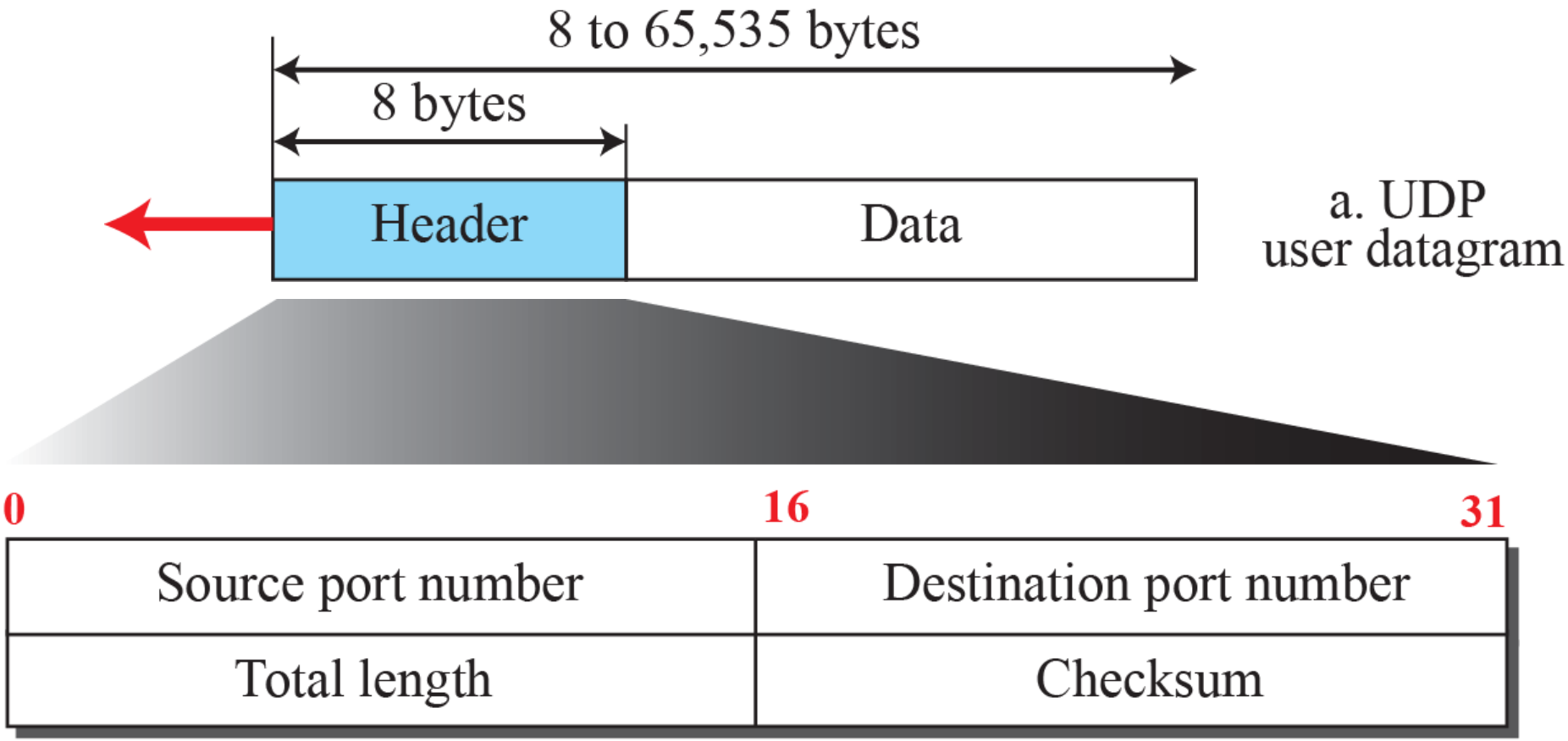


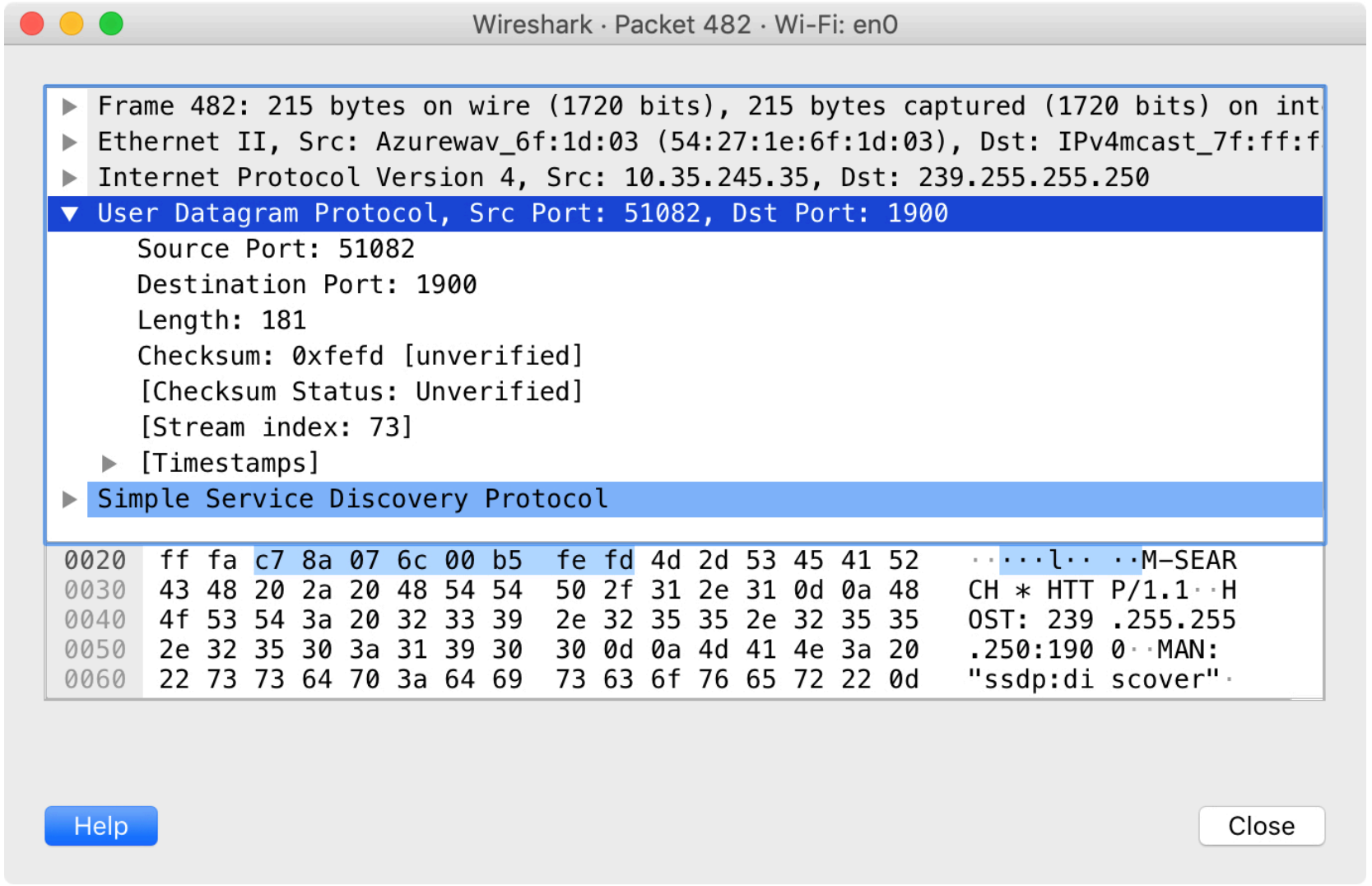
Well-known service ports

Registered ports

Ephemeral ports (dynamic, private, temporary)

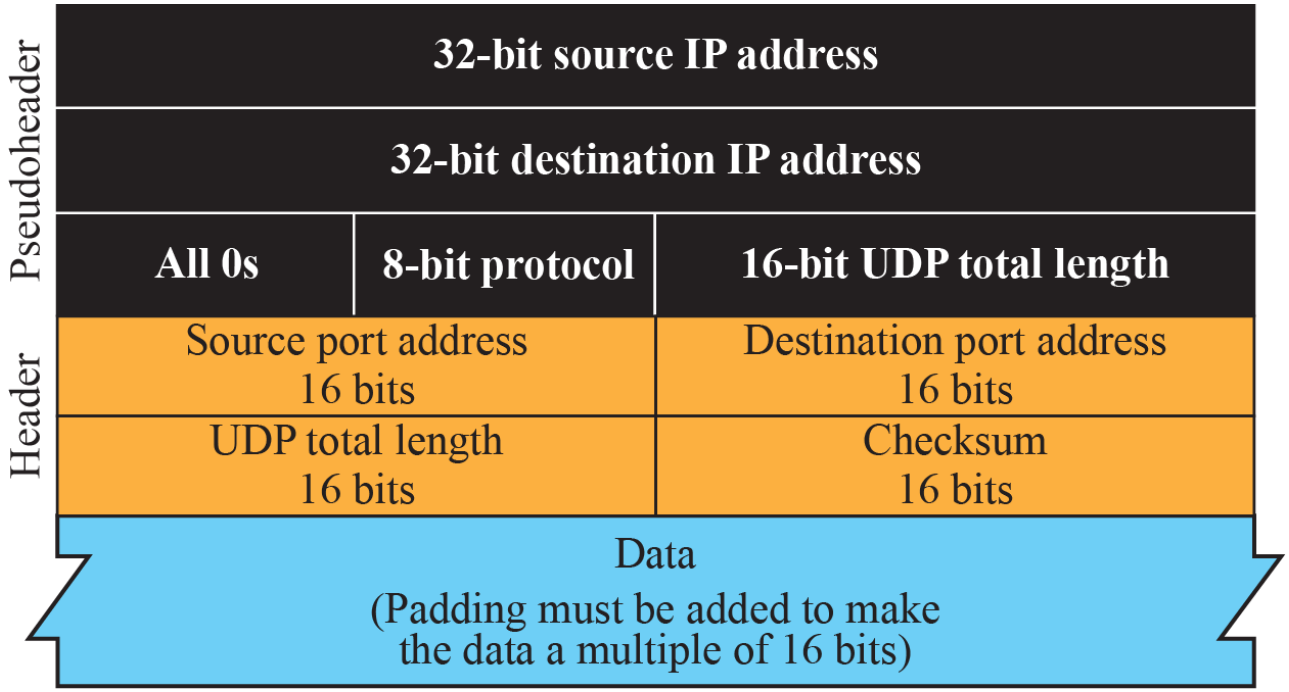
	Local port	Remote port
Client side	Random from Ephemeral range	Service or Registered port
Server side	Service or Registered port	Derived from client segment


8 to 65,535 bytes
8 bytes
Header
Data
a. UDP
user datagram
0
16
31
Source port number
Destination port number
Total length
Checksum
b. Header format


Wireshark · Packet 482 · Wi-Fi: en0
Frame 482: 215 bytes on wire (1720 bits), 215 bytes captured (1720 bits) on int
Ethernet II, Src: Azurewav_6f:1d:03 (54:27:1e:6f:1d:03), Dst: IPv4mcast_7f:ff:f
Internet Protocol Version 4, Src: 10.35.245.35, Dst: 239.255.255.250
User Datagram Protocol, Src Port: 51082, Dst Port: 1900
Source Port: 51082
Destination Port: 1900
Length: 181
Checksum: 0xfefd [unverified]
[Checksum Status: Unverified]
[Stream index: 73]
[Timestamps]
Simple Service Discovery Protocol
0020 ff fa c7 8a 07 6c 00 b5 fe fd 4d 2d 53 45 41 52 ·····l·· ··M-SEAR
0030 43 48 20 2a 20 48 54 54 50 2f 31 2e 31 0d 0a 48 CH * HTT P/1.1··H
0040 4f 53 54 3a 20 32 33 39 2e 32 35 35 2e 32 35 35 OST: 239 .255.255
0050 2e 32 35 30 3a 31 39 30 30 0d 0a 4d 41 4e 3a 20 .250:190 0··MAN:
0060 22 73 73 64 70 3a 64 69 73 63 6f 76 65 72 22 0d "ssdp:di scover"·
Help
Close

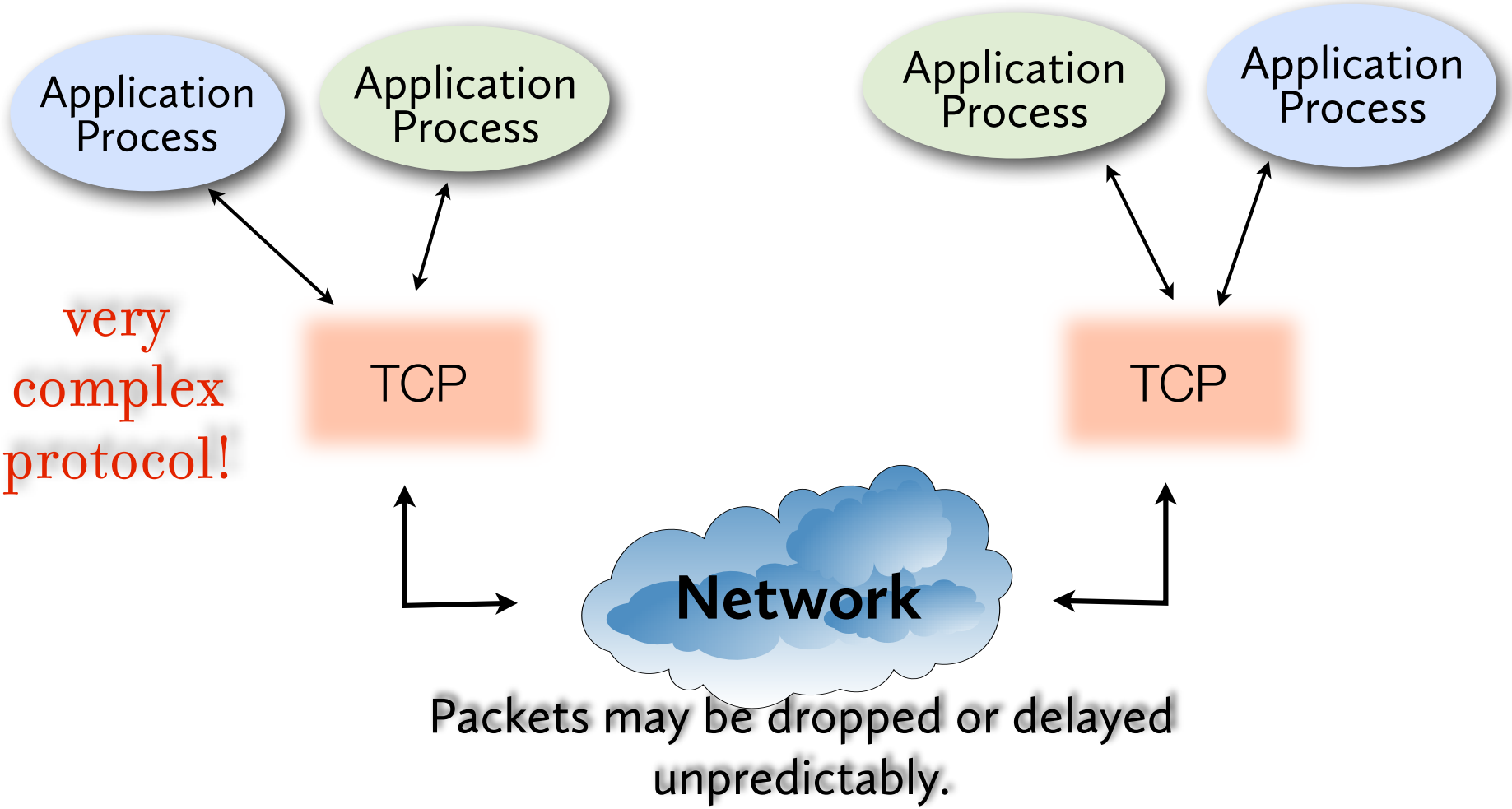
UDP (Internet) Checksum

```
  1  1  1  0  0  1  1  0  0  1  1  0  0  1  1  0
  1  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1

(1) 1  0  1  1  1  0  1  1  1  0  1  1  1  0  1  1

  1  0  1  1  1  0  1  1  1  0  1  1  1  0  1  1
  0  1  0  0  0  1  0  0  0  1  0  0  0  1  0  0
```

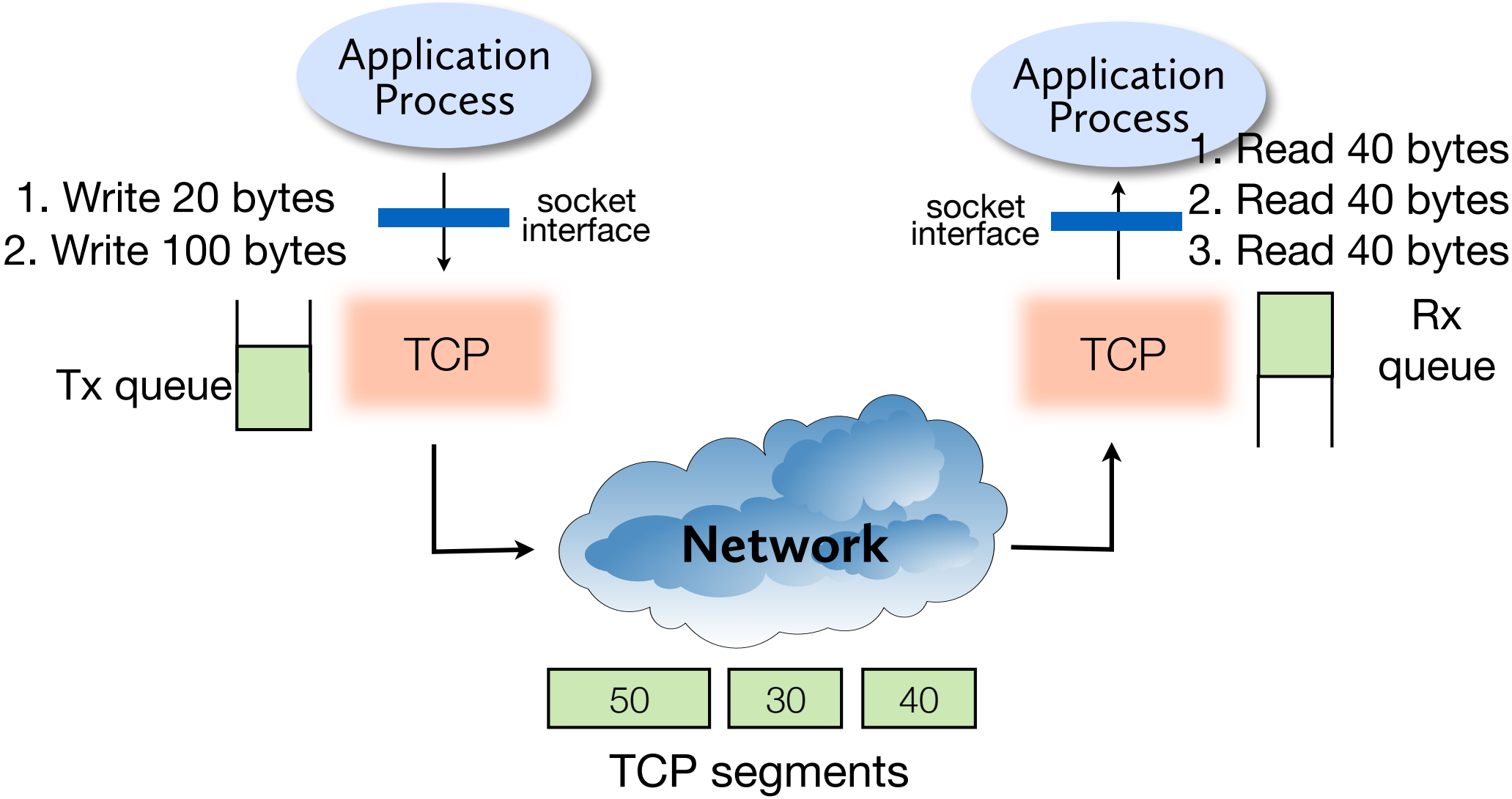


TCP: Connection-Oriented Transport

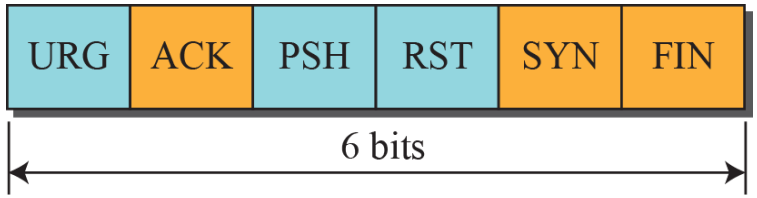
- **Reliable, full-duplex, byte-stream delivery** over an unreliable network.



Byte-Stream Service

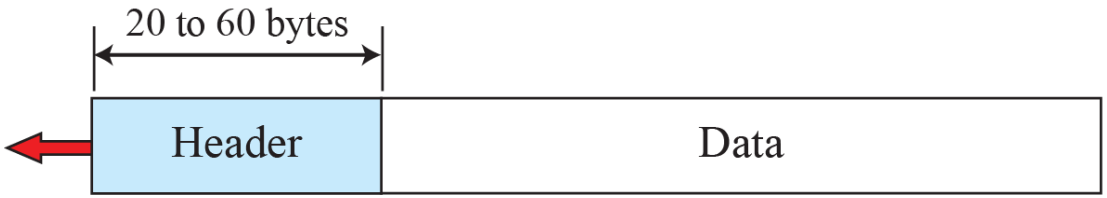


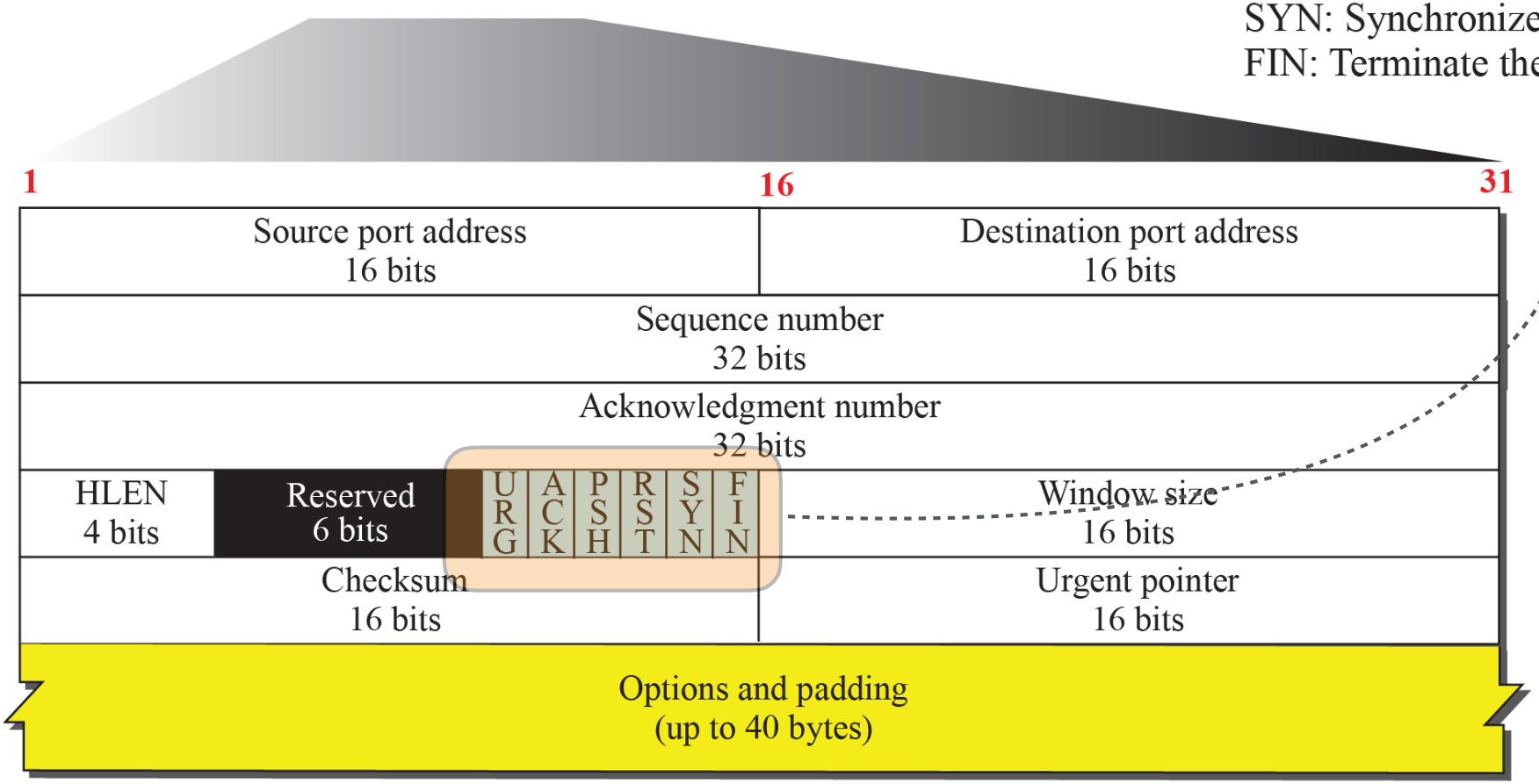
TCP Segment Format



URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

a



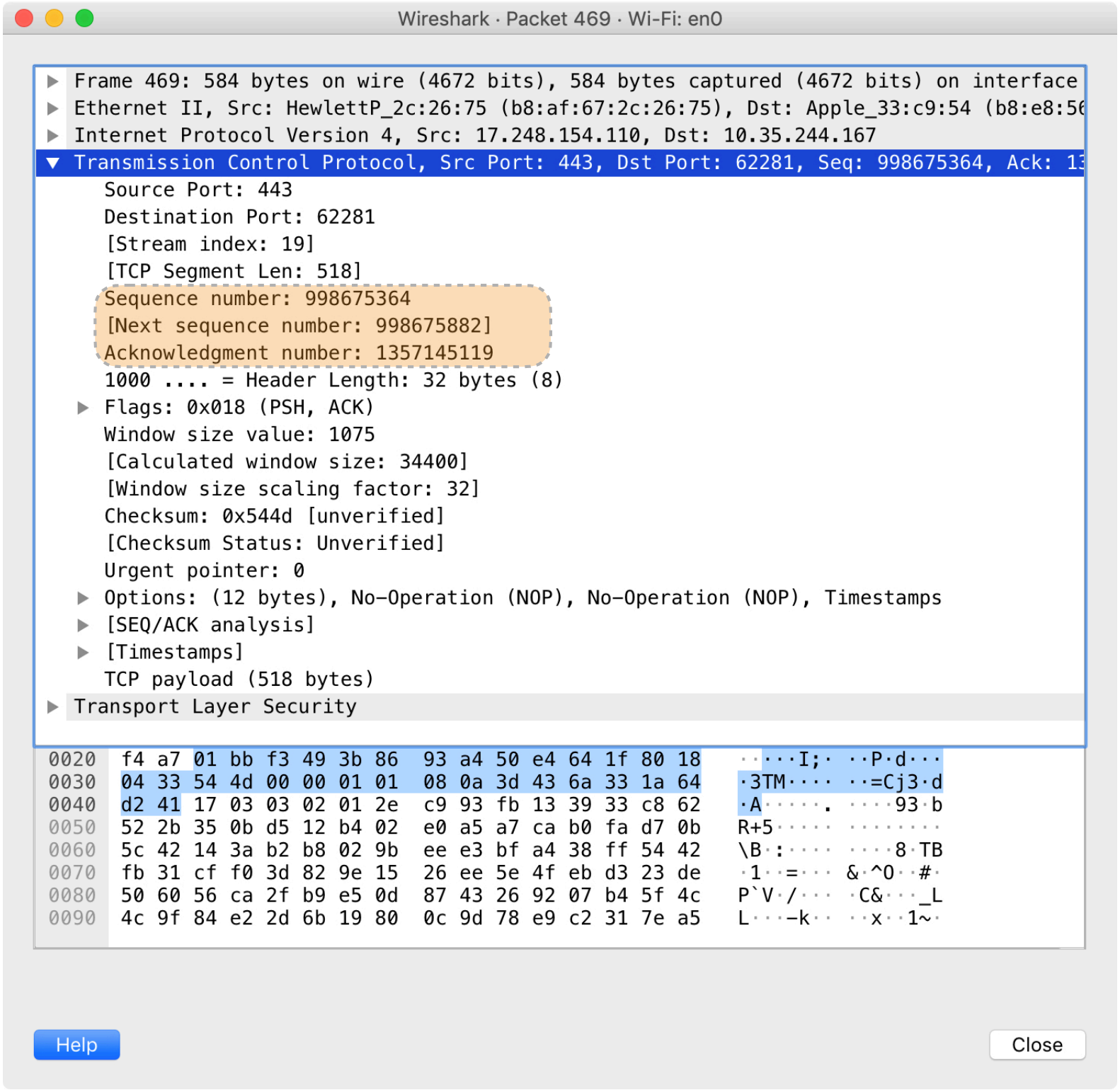


b. Header

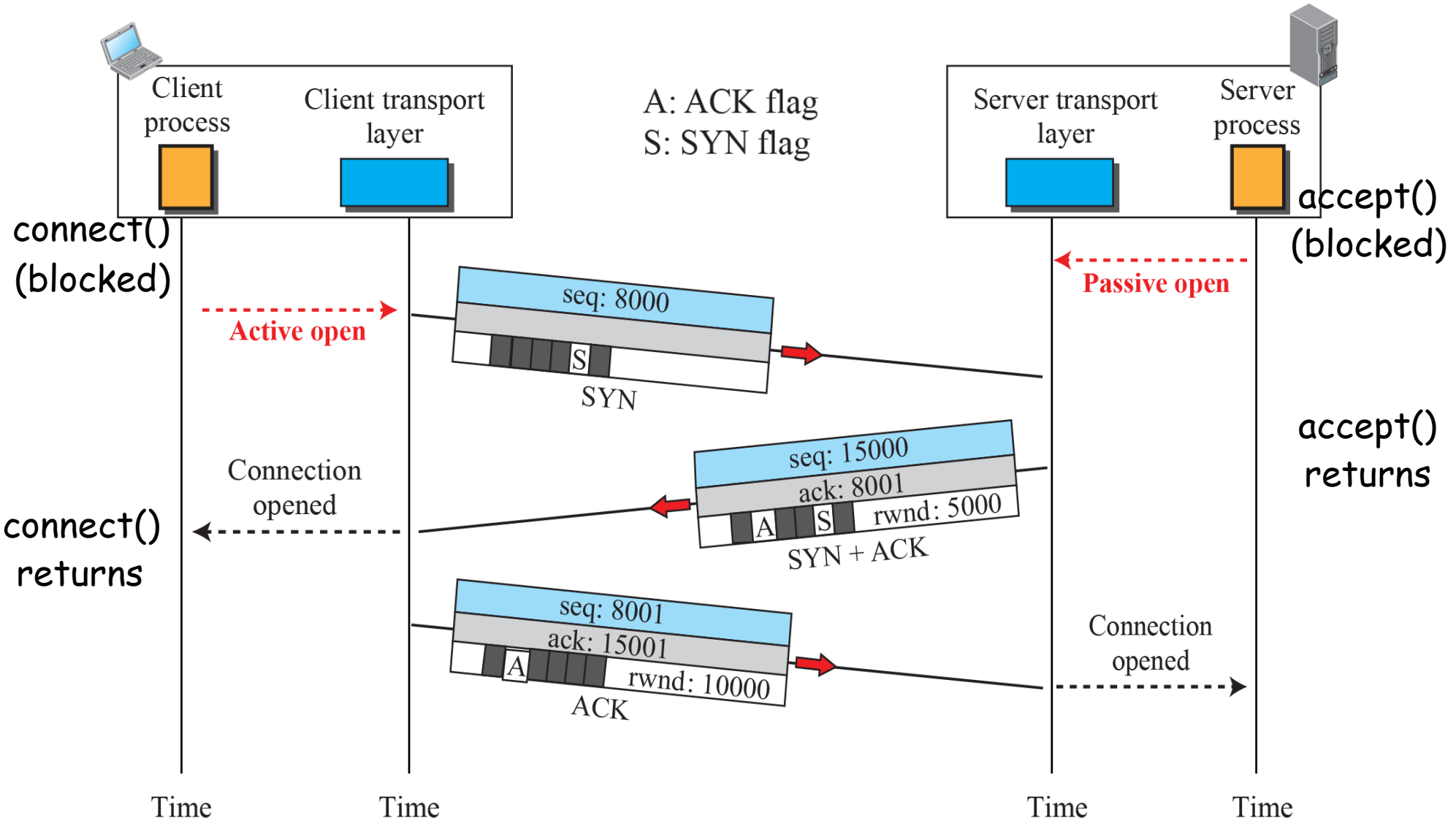
TCP Byte Numbering and Sequence No.

- Number bytes sequentially. Why?
- Segment seq. no. = Seq. no of 1st byte in segment
- Ex: Send 5000-byte file in 5 segments, starting with byte number 10001.

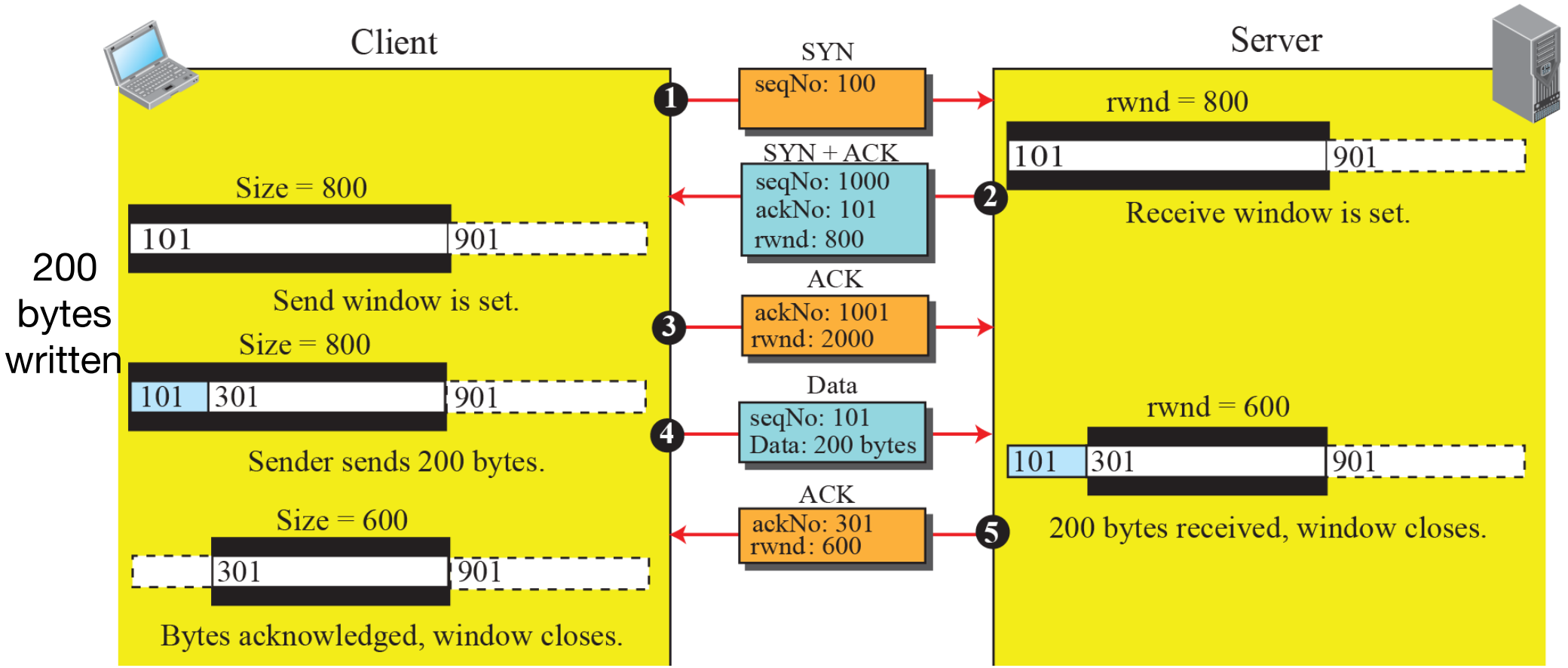
Segment 1	→	Sequence Number:	10,001	**Range:**	10,001	to	11,000
Segment 2	→	Sequence Number:	11,001	**Range:**	11,001	to	12,000
Segment 3	→	Sequence Number:	12,001	**Range:**	12,001	to	13,000
Segment 4	→	Sequence Number:	13,001	**Range:**	13,001	to	14,000
Segment 5	→	Sequence Number:	14,001	**Range:**	14,001	to	15,000


Wireshark · Packet 469 · Wi-Fi: en0
Frame 469: 584 bytes on wire (4672 bits), 584 bytes captured (4672 bits) on interface
Ethernet II, Src: HewlettP_2c:26:75 (b8:af:67:2c:26:75), Dst: Apple_33:c9:54 (b8:e8:56
Internet Protocol Version 4, Src: 17.248.154.110, Dst: 10.35.244.167
Transmission Control Protocol, Src Port: 443, Dst Port: 62281, Seq: 998675364, Ack: 13
Source Port: 443
Destination Port: 62281
[Stream index: 19]
[TCP Segment Len: 518]
Sequence number: 998675364
[Next sequence number: 998675882]
Acknowledgment number: 1357145119
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
Window size value: 1075
[Calculated window size: 34400]
[Window size scaling factor: 32]
Checksum: 0x544d [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (518 bytes)
Transport Layer Security
0020 f4 a7 01 bb f3 49 3b 86 93 a4 50 e4 64 1f 80 18
0030 04 33 54 4d 00 00 01 01 08 0a 3d 43 6a 33 1a 64
0040 d2 41 17 03 03 02 01 2e c9 93 fb 13 39 33 c8 62
0050 52 2b 35 0b d5 12 b4 02 e0 a5 a7 ca b0 fa d7 0b
0060 5c 42 14 3a b2 b8 02 9b ee e3 bf a4 38 ff 54 42
0070 fb 31 cf f0 3d 82 9e 15 26 ee 5e 4f eb d3 23 de
0080 50 60 56 ca 2f b9 e5 0d 87 43 26 92 07 b4 5f 4c
0090 4c 9f 84 e2 2d 6b 19 80 0c 9d 78 e9 c2 31 7e a5
Help
Close

TCP Connection Establishment: 3-Way Handshake



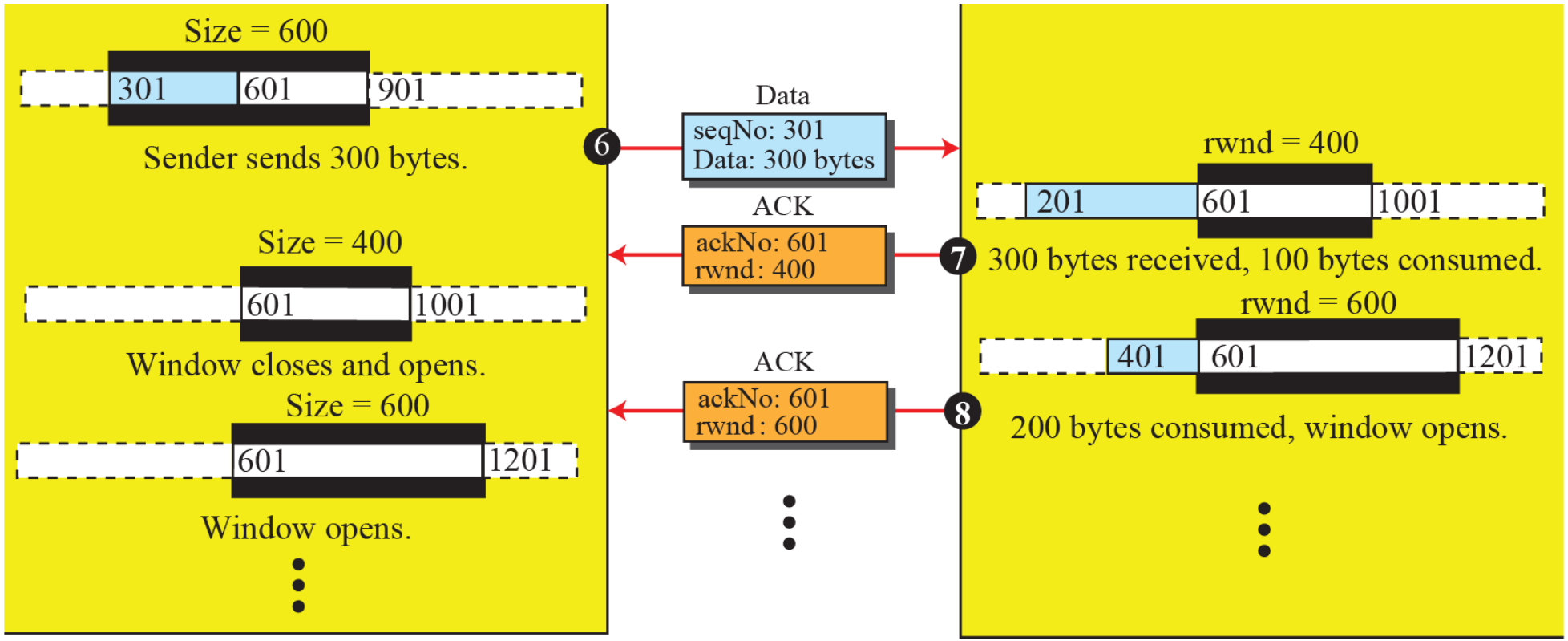
Rx buffer = 800 bytes



rwnd = Receive buffer space available

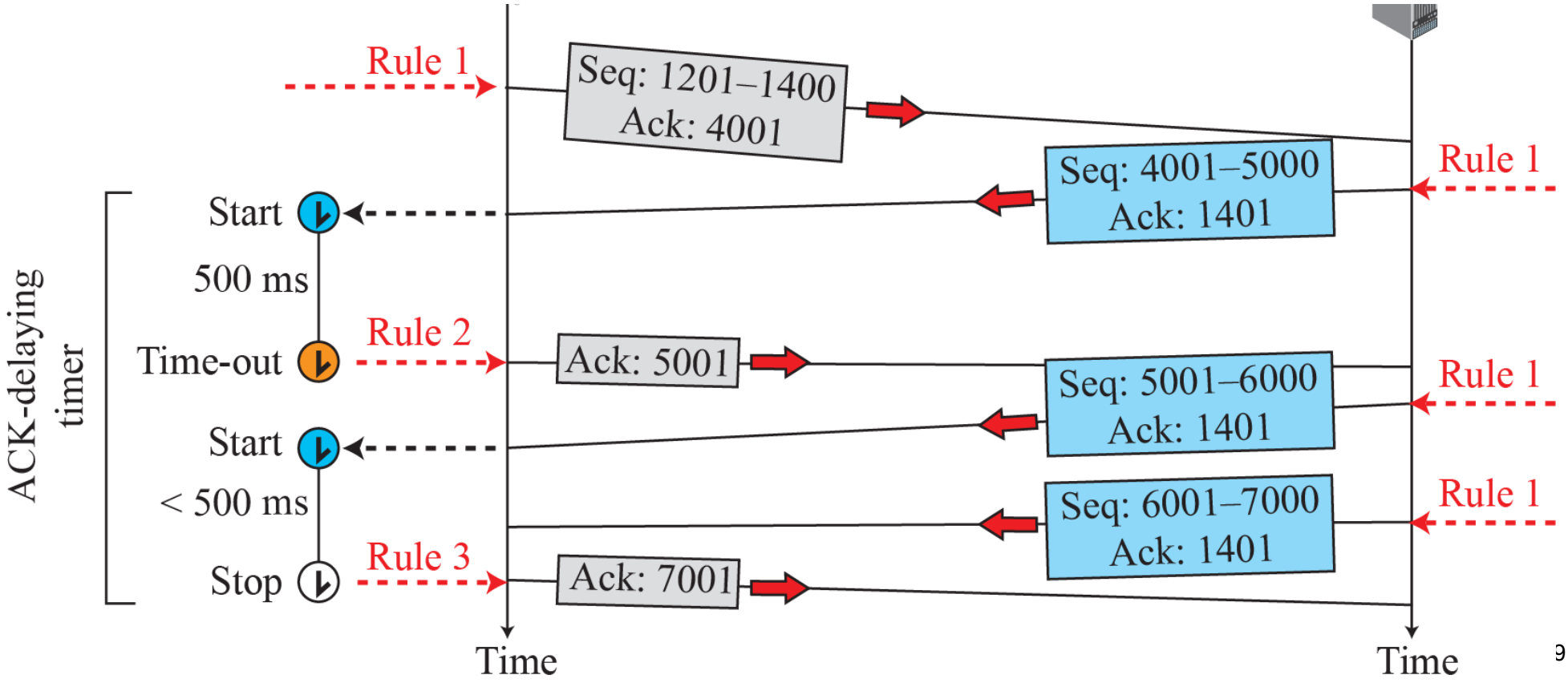
Receiving side keeps # bytes to send less than rwnd

Another 300 bytes written

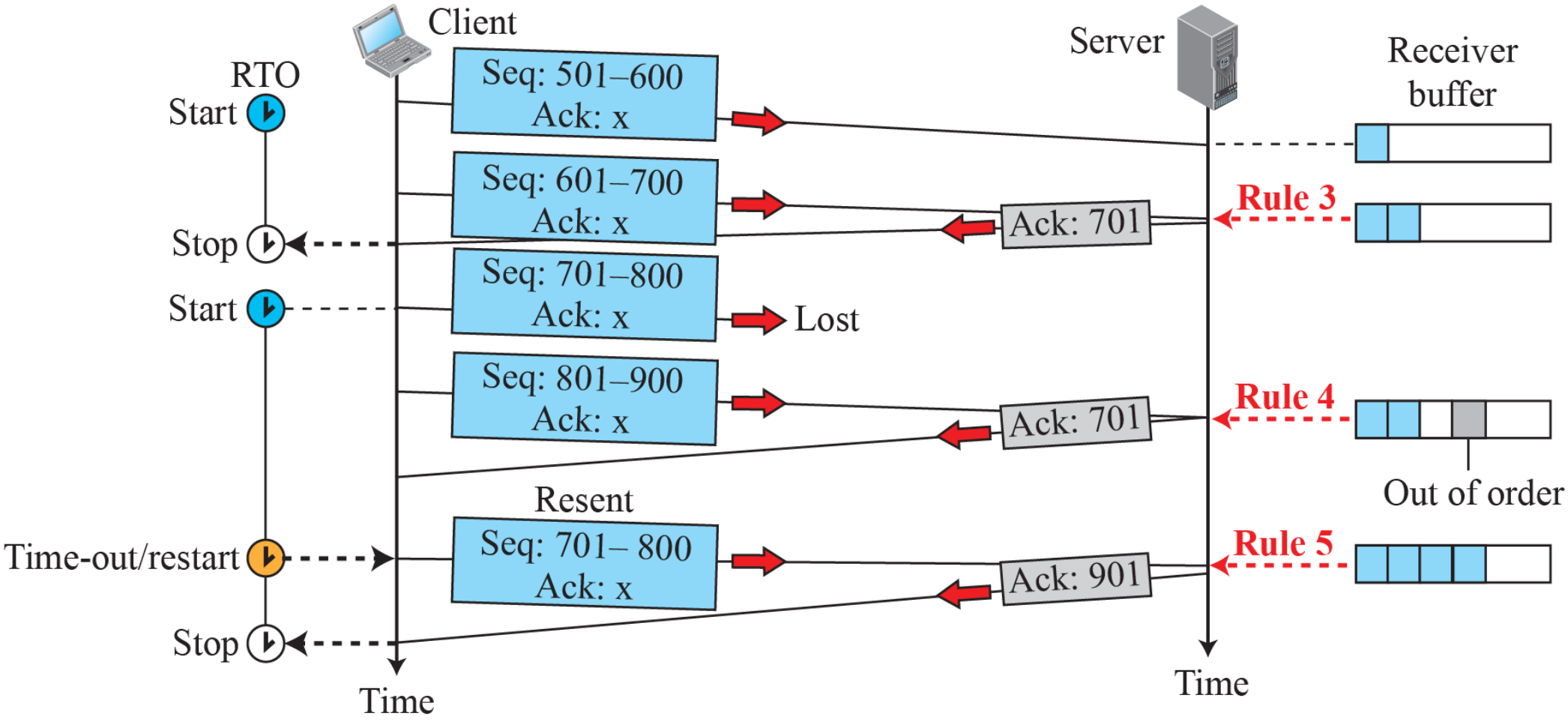


TCP Error Control -- Normal Operation

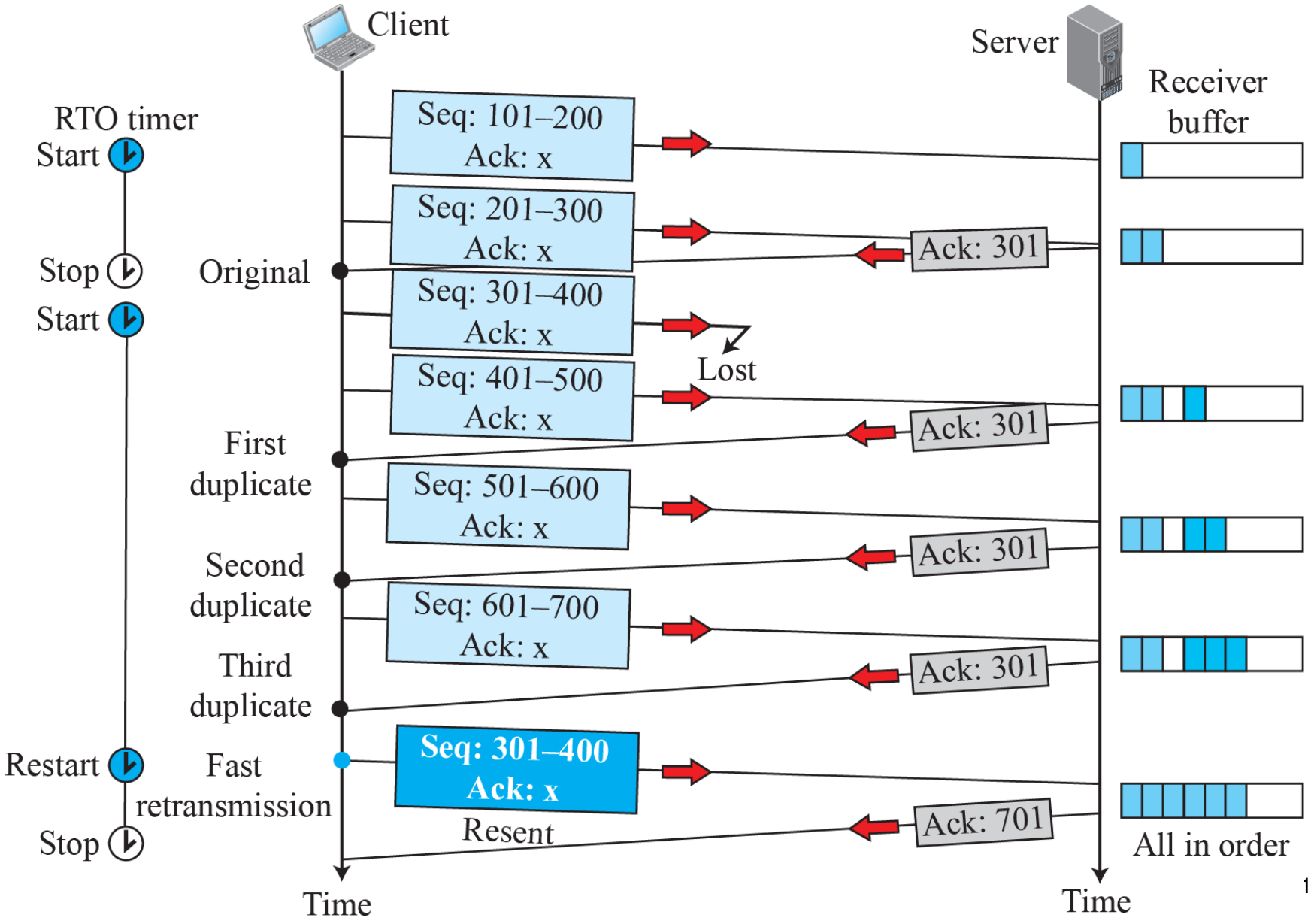
- See Acknowledgment rules in p.206 of Forouzan text
- Only one timer for the first outstanding segment.



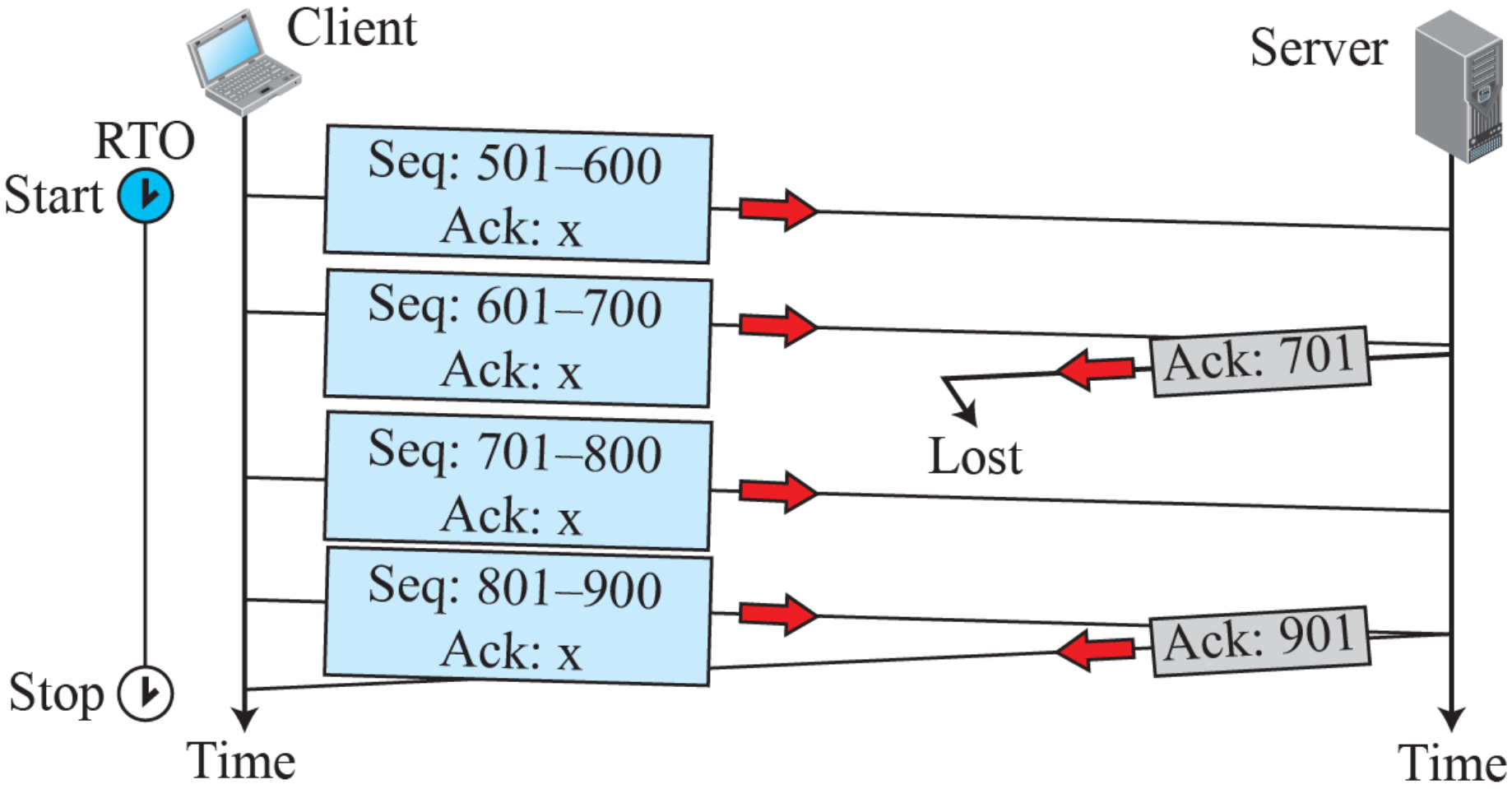
Lost Segment

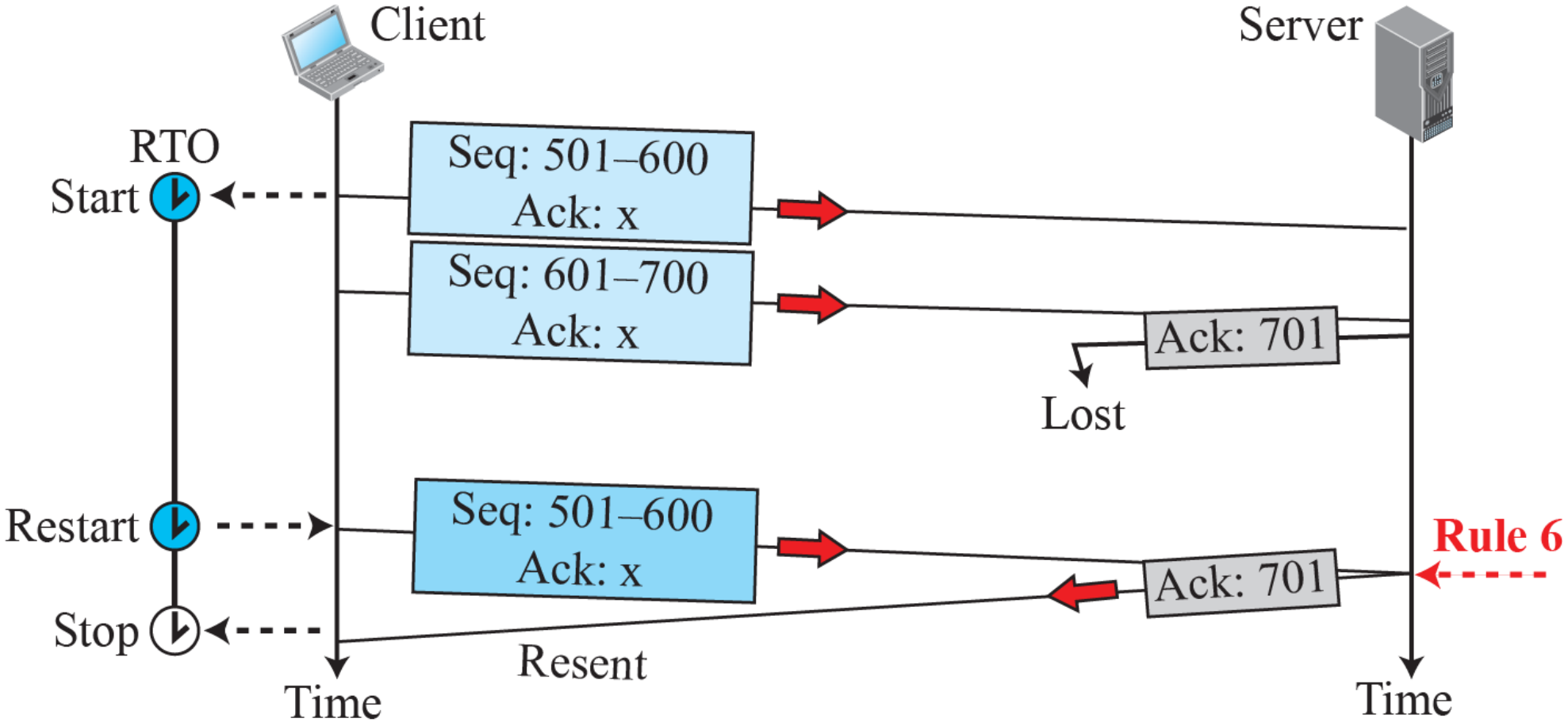


Fast Retransmission

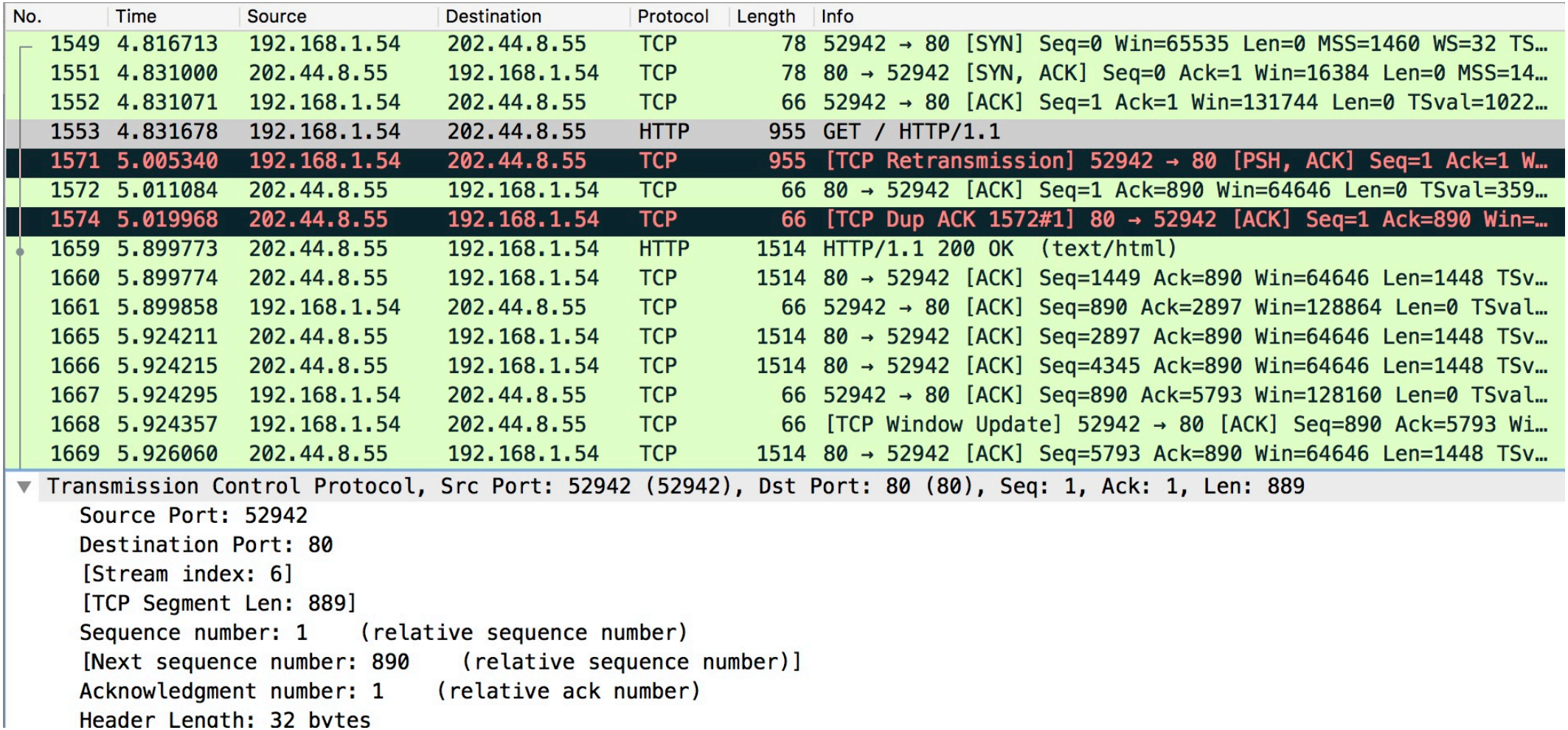


Lost Acknowledgment




Client
Server
RTO
Start
Seq: 501–600
Ack: x
Seq: 601–700
Ack: x
Ack: 701
Lost
Restart
Seq: 501–600
Ack: x
Rule 6
Ack: 701
Stop
Resent
Time
Time

In Wireshark preference, go to protocols->tcp and checkbox the option 'Relative sequence number'



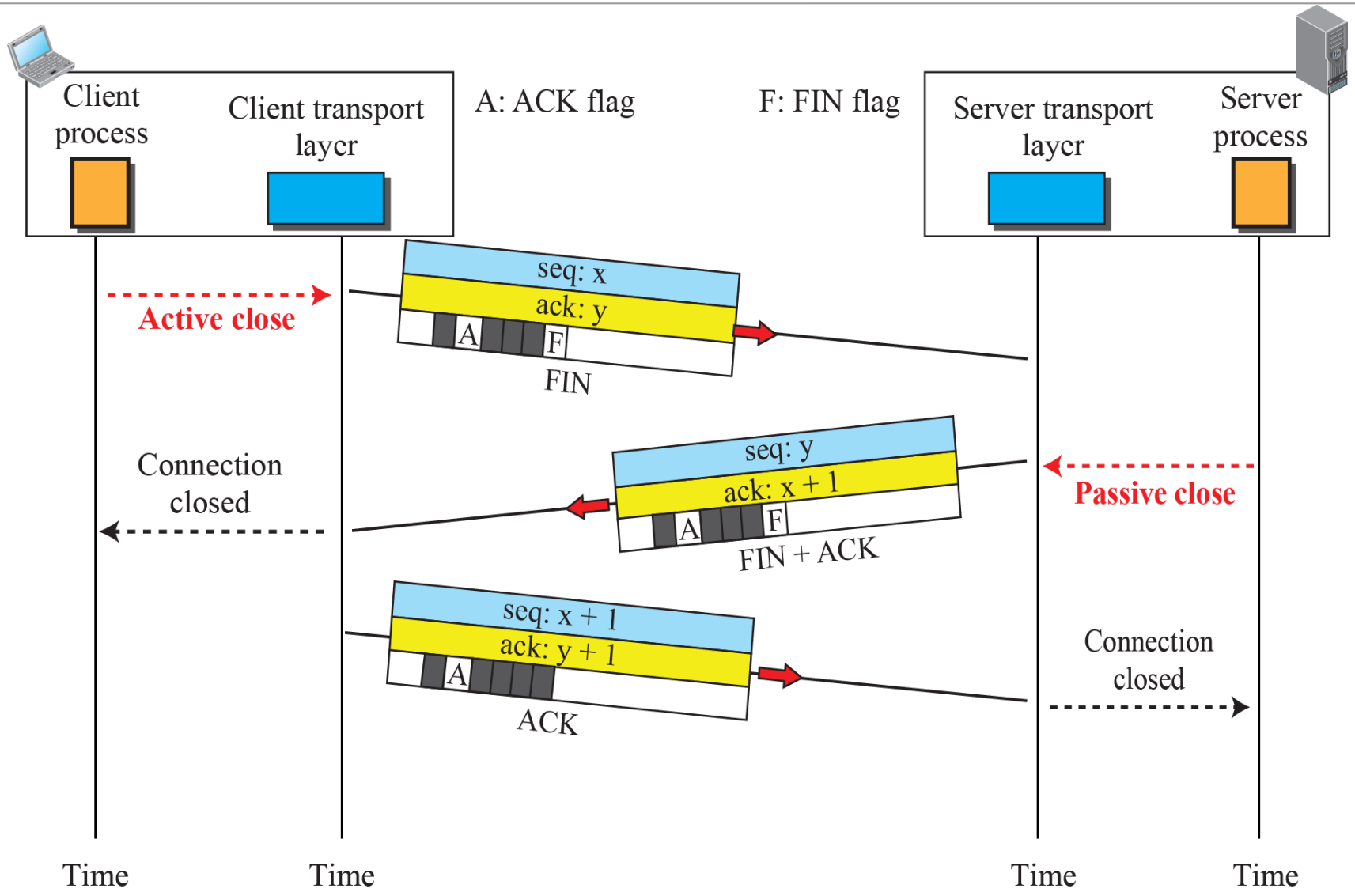
Self-Test

- How long you application need to wait before it can send the first message to the destination after opening a TCP connection ?
  A. One Round-Trip-Time (RTT)
  B. One-and-a-half RTTs
  C. Two RTTs
  D. Four RTTs

Self-Test

- Suppose TCP continously sends segments for your application. If one of the segments has lost but all the successive segments get through (before the RTO timer expires), approximately how long will it take for your application to receive the lost segment ?
  A. One RTT
  B. Two RTTs
  C. Three RTTs
  D. Four RTTs

TCP Connection Teardown



Exploiting TCP Connection Management

- TCP SYN scan
  - Open a TCP connection to each port number one by one.
  - SYN/ACK returned for open ports, RST for closed ports
- TCP FIN scan
  - Send a FIN to each port number one by one.
  - Nothing returned for open ports, RST for closed ports
- TCP SYN DoS attack
  - Send SYNs to establish connections but doesn't return ACK.
  - The victim runs out of memory and possibly crashes.

Activity I: Which Transport Protocol for My App ?

- Suppose you are to choose transport protocol (UDP and TCP) for your network application.
  - List and explain possible criteria that you will use to make the decision.
  - Can you prioritize the list ?

Conclusion

- End-to-end layer for process multiplexing, flow and error control, congestion control
  - Unreliable vs. Reliable transport service
  - Connectionless vs. Connection-oriented service
- TCP operations
  - Connection set up and teardown phases -- State initialization and resource management
  - Sliding window flow control and error control
  - High overhead for short transactions apps