

CS118 Discussion 1C, Week 4

Zengwen Yuan

Bunche Hall 3156, Friday 2:00—3:50 p.m.

Outline

- Lecture review: Transport layer
- Examples
- Midterm logistics

Stop-and-Wait Protocol

- It's rdt-3.0 protocol in the textbook
 - sender waits “reasonable” amount of time for ACK
 - don't send until hear the ACK
 - retransmit upon timeout
- Main Issue: **limited performance**

Stop-and-Wait Protocol

- Consider two hosts that are directly connected by a 50 Kbit/sec satellite link that has a 250 milliseconds propagation delay. If these hosts send 1000 bits segments, what is the maximum throughput in stop-and-wait protocol if we ignore the transmission time of ACK?
- $1000 / (1000 / 50000 + 0.25 + 0.25) = 1923 \text{ bps} < \mathbf{2 \text{ Kbps}} \ll \mathbf{50 \text{ Kbps!}}$

Pipelined Protocols

- Go-back-N: receiver only sends **cumulative** ACKs
 - Drop out-of-order segments
 - reACK packet with highest in-order sequence number
 - Timer for oldest unACKed packet only, retransmit all unACKed packets
- Selective repeat: receiver ACKs **individual packets (in class we used cumulative ACK, similar to TCP's choice)**
 - Buffer out of order segments
 - Timer for each individual unACKed packet, retransmit any unACKed packet

Demo: Selective Repeat/Go Back N

- http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

Selective Repeat / Go Back N

configuration

protocol

- Go back N
- Selective Repeat

choosing a new protocol restarts the simulation

window size

sets the window size for the windows

end to end delay

time a packet takes from one station to the other

timeout

scroll mode

Typewriter style

change the style the window scrolls

automatic emission of packets

stop

starts or stops the automatic emission of packets by the upper layer

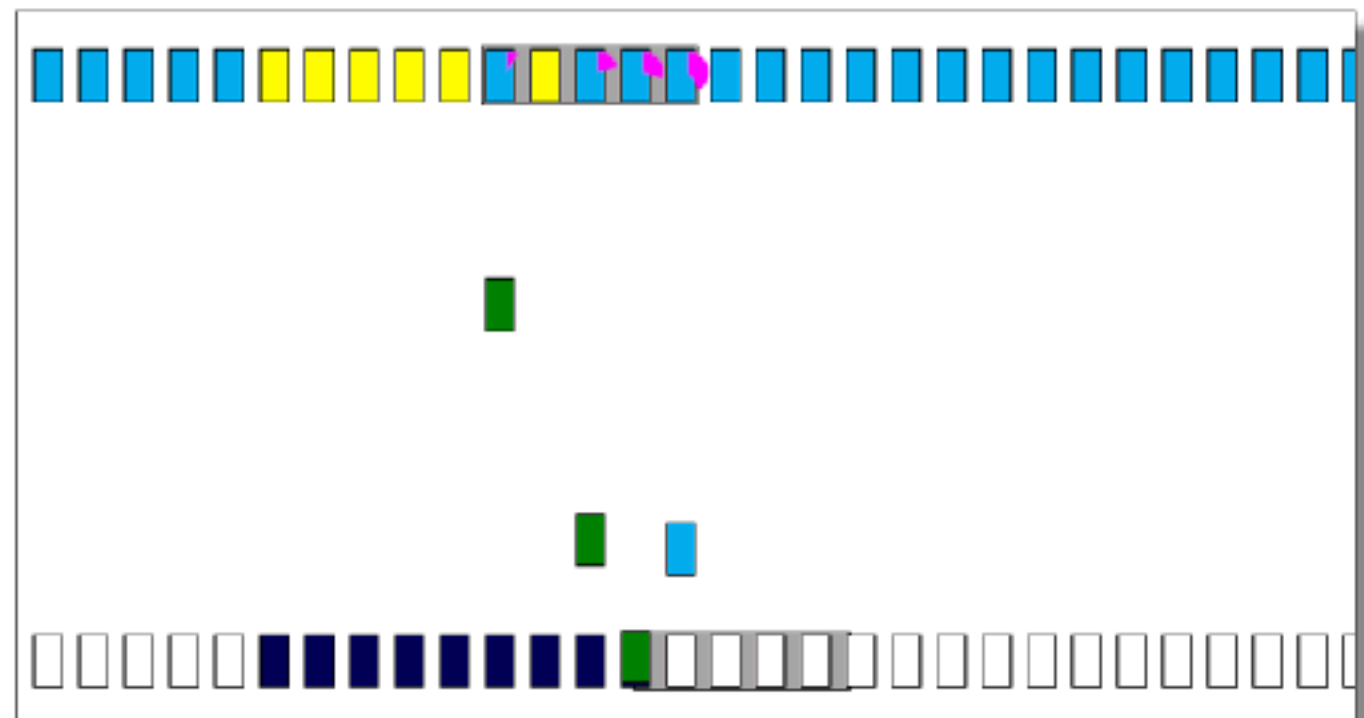
number of packets emitted per minute

60

the number of packets the upper layer tries to send per minute

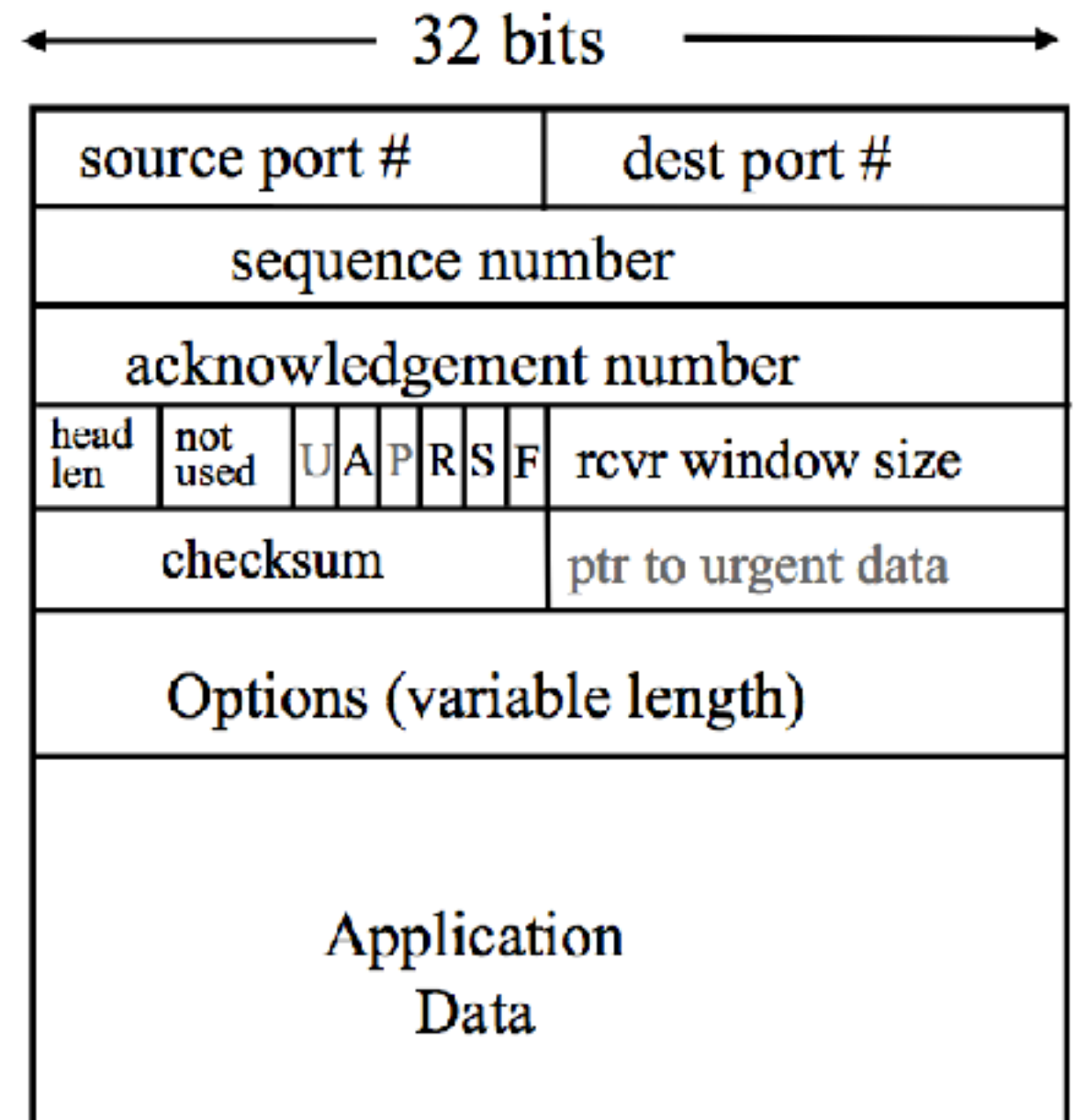
legend

- no data received yet
- data buffered (ready to send, delivered or sent but no ack received yet)
- ack
- transmission confirmed
- data has been delivered to upper network layer



TCP

- Point-to-point, byte-stream reliable transport protocol
 - **Multiplexing/de-multiplexing:** Source/Dest port
 - **Reliable data transfer:** sequence number, ack, checksum, RTT estimation
 - **Connection setup:** sequence number, SYN, receive window
 - **Connection teardown:** sequence number, FIN



Comparison of Reliable Transport Protocol

Protocol	Buffer at sender	Buffer at receiver	ACK	Timeout/Retransmission
Stop & Wait	No	No	No out-of-order	Retransmit timeout packet
Go-Back-N	Yes	No	Cumulative Seq#	Retransmit all packets in window
Selective Repeat	Yes	Yes	Received Seq#	Retransmit timeout packet
TCP	Yes	Yes	Next-expected cumulative Seq#	Retransmit timeout packet

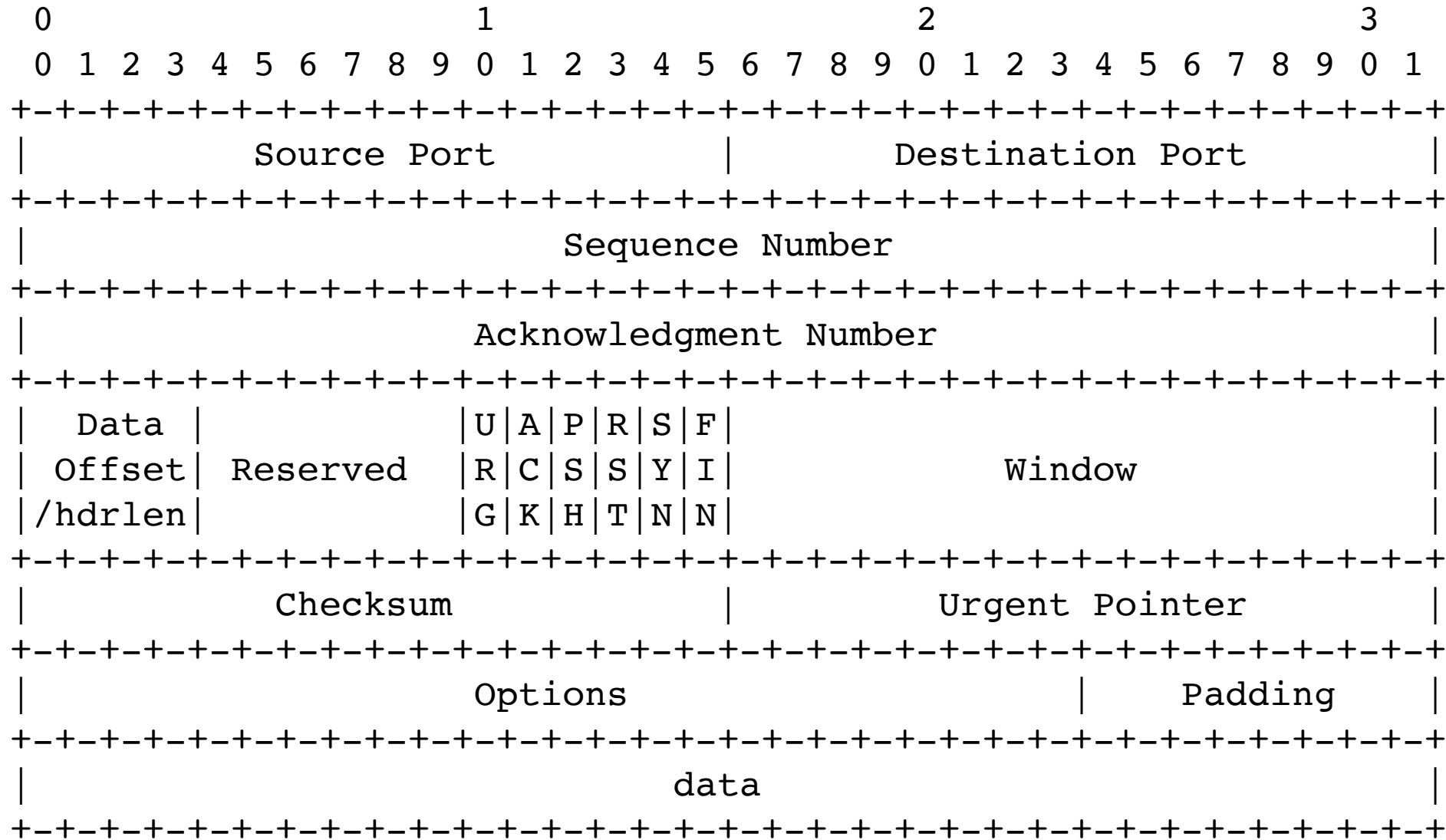
TCP: overview

- The Transmission Control Protocol (TCP), documented in RFC 793, makes up for IP's deficiencies by providing reliable, stream-oriented connections that hide most of IP's shortcomings. The protocol suite gets its name because most TCP/IP protocols are based on TCP, which is in turn based on IP. TCP and IP are the twin pillars of TCP/IP.

TCP: key functionalities

- TCP adds a great deal of functionality to the IP service it is layered over:
 - **Streams.** TCP data is organized as a stream of bytes, much like a file. The datagram nature of the network is concealed. A mechanism (the Urgent Pointer) exists to let out-of-band data be specially flagged.
 - **Reliable delivery.** Sequence numbers are used to coordinate which data has been transmitted and received. TCP will arrange for retransmission if it determines that data has been lost.
 - **Network adaptation.** TCP will dynamically learn the delay characteristics of a network and adjust its operation to maximize throughput without overloading the network.
 - **Flow control.** TCP manages data buffers, and coordinates traffic so its buffers will never overflow. Fast senders will be stopped periodically to keep up with slower receivers.

TCP: header format



TCP Header Format

Note that one tick mark represents one bit position.

TCP: header explained

- Source Port: 16 bits
 - The source port number.
- Destination Port: 16 bits
 - The destination port number.
- Sequence Number: 32 bits
 - The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.
- Acknowledgment Number: 32 bits
 - If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.
- Data Offset: 4 bits
 - The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.

TCP: header explained (cont'd)

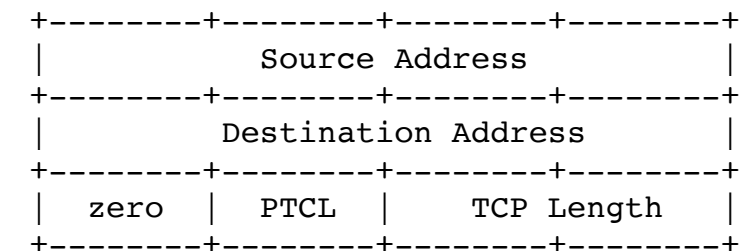
- Reserved: 6 bits
 - Reserved for future use. Must be zero.
- Control Bits: 6 bits (from left to right):
 - URG: Urgent Pointer field significant
 - ACK: Acknowledgment field significant
 - PSH: Push Function
 - RST: Reset the connection
 - SYN: Synchronize sequence numbers
 - FIN: No more data from sender
- Window: 16 bits
 - The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

TCP: header explained (cont'd)

- Checksum: 16 bits

- The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

- The checksum also covers a 96 bit pseudo header conceptually prefixed to the TCP header. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in the arguments or results of calls by the TCP on the IP. The TCP Length is the TCP header length plus the data length in octets (this is not an explicitly transmitted quantity, but is computed), and it does not count the 12 octets of the pseudo header.



- Urgent Pointer: 16 bits

- This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set.

TCP: connection setup

- Connection setup: three-way handshaking
 - 1st round: SYN+initial sequence number
 - 2nd round: SYN+SYNACK+server's initial sequence number
 - 3rd round: SYNACK+(optional) data

TCP: connection setup

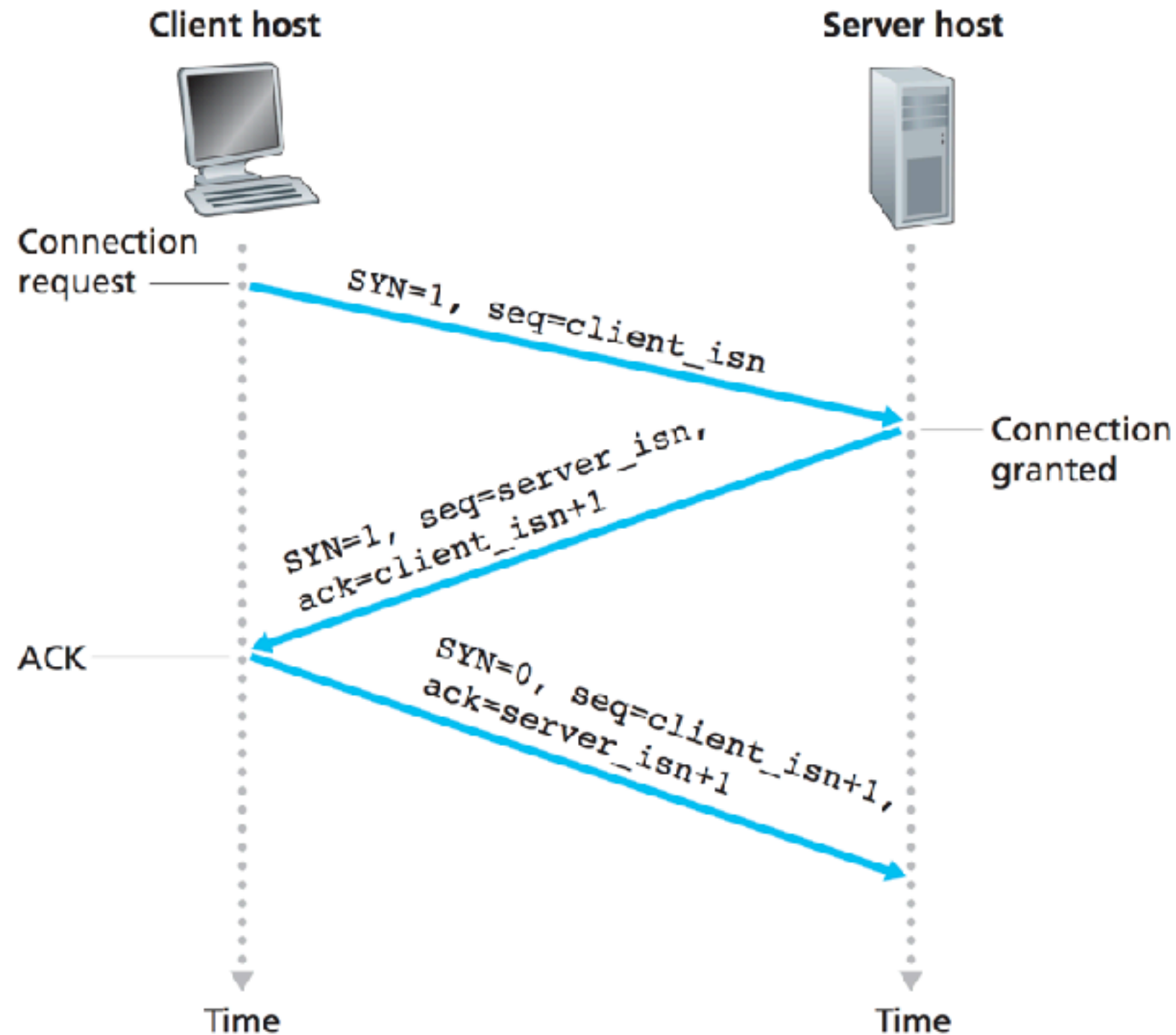
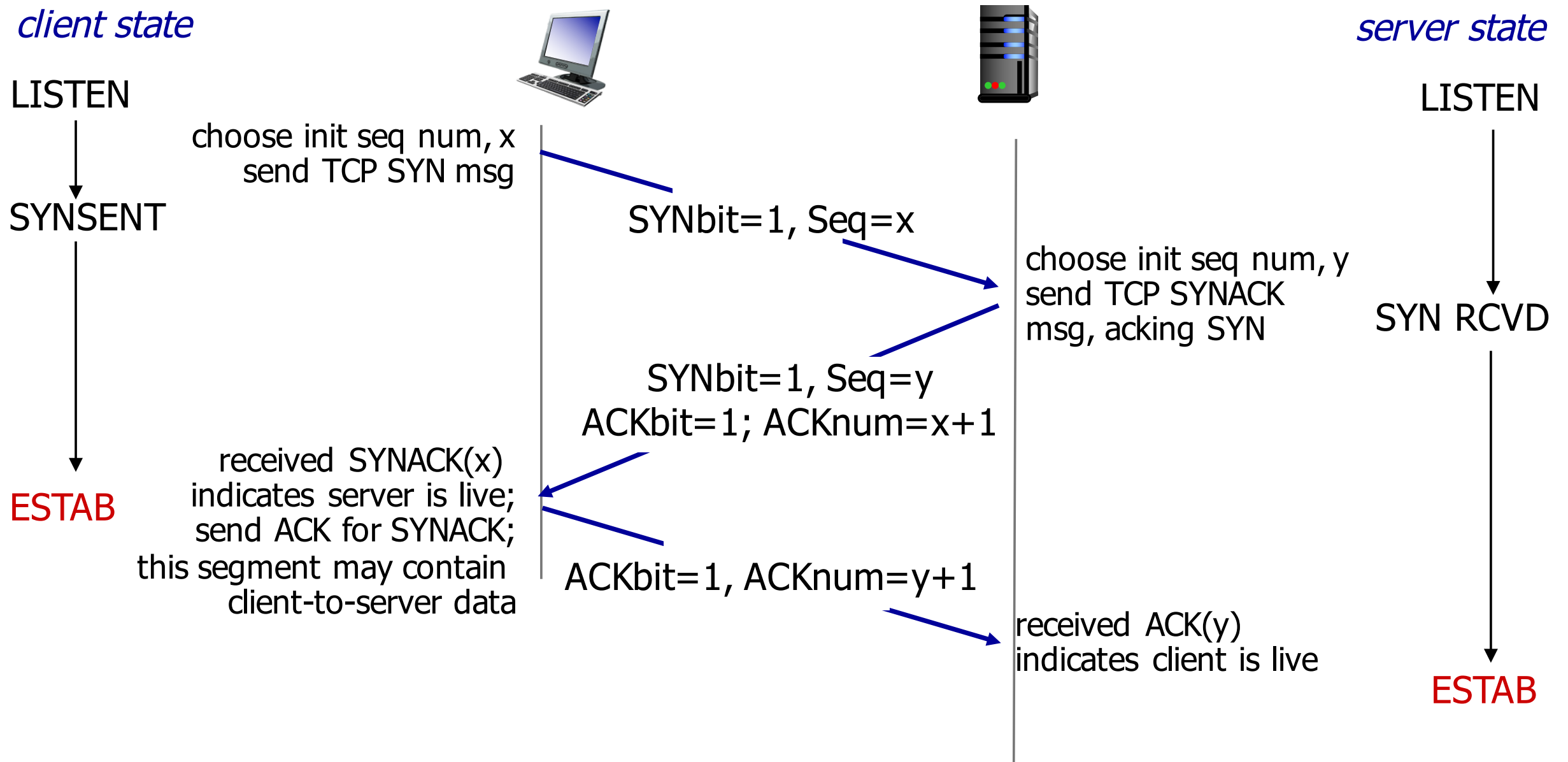


Figure 3.39 ♦ TCP three-way handshake: segment exchange

TCP: connection setup (cont'd)

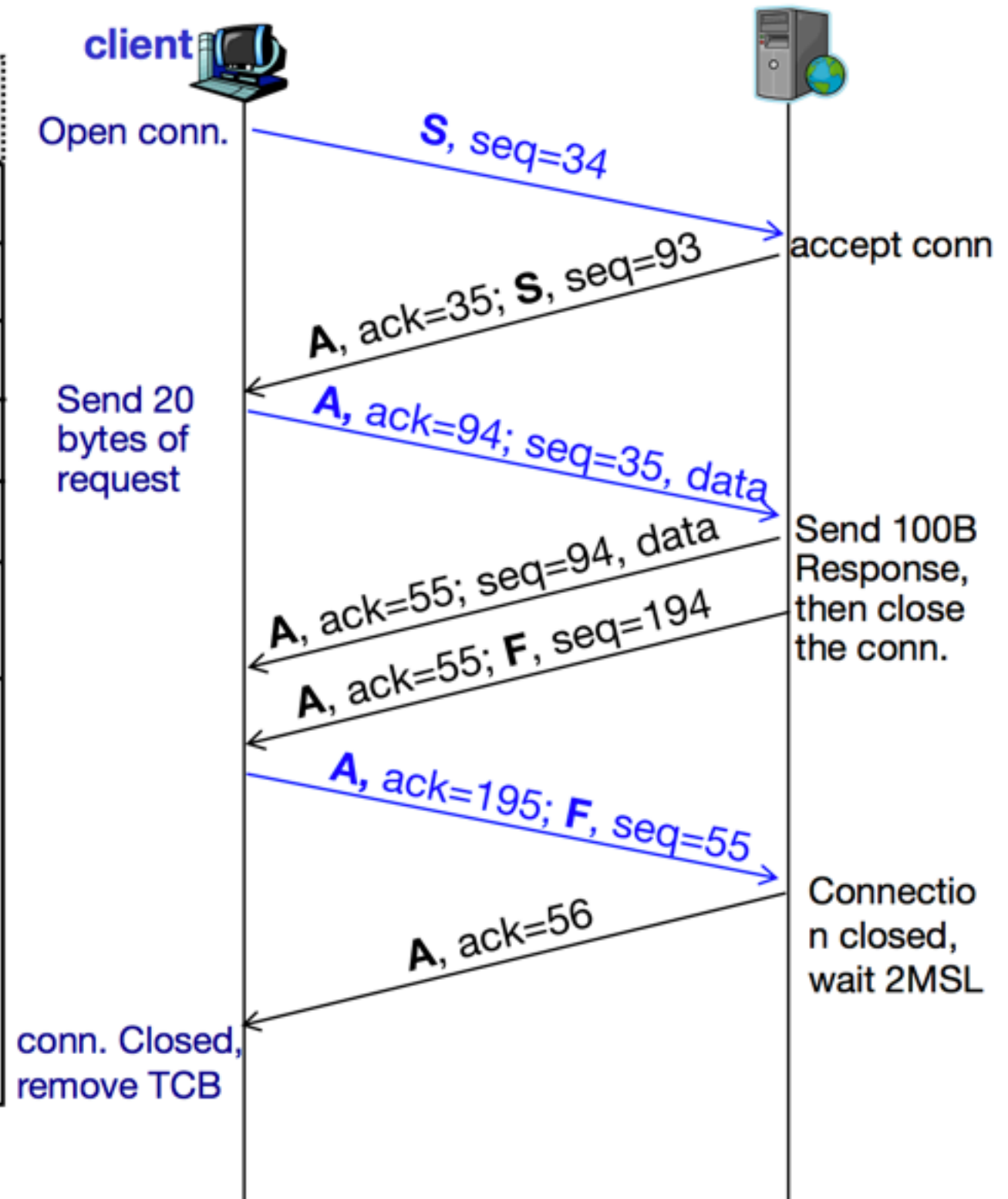
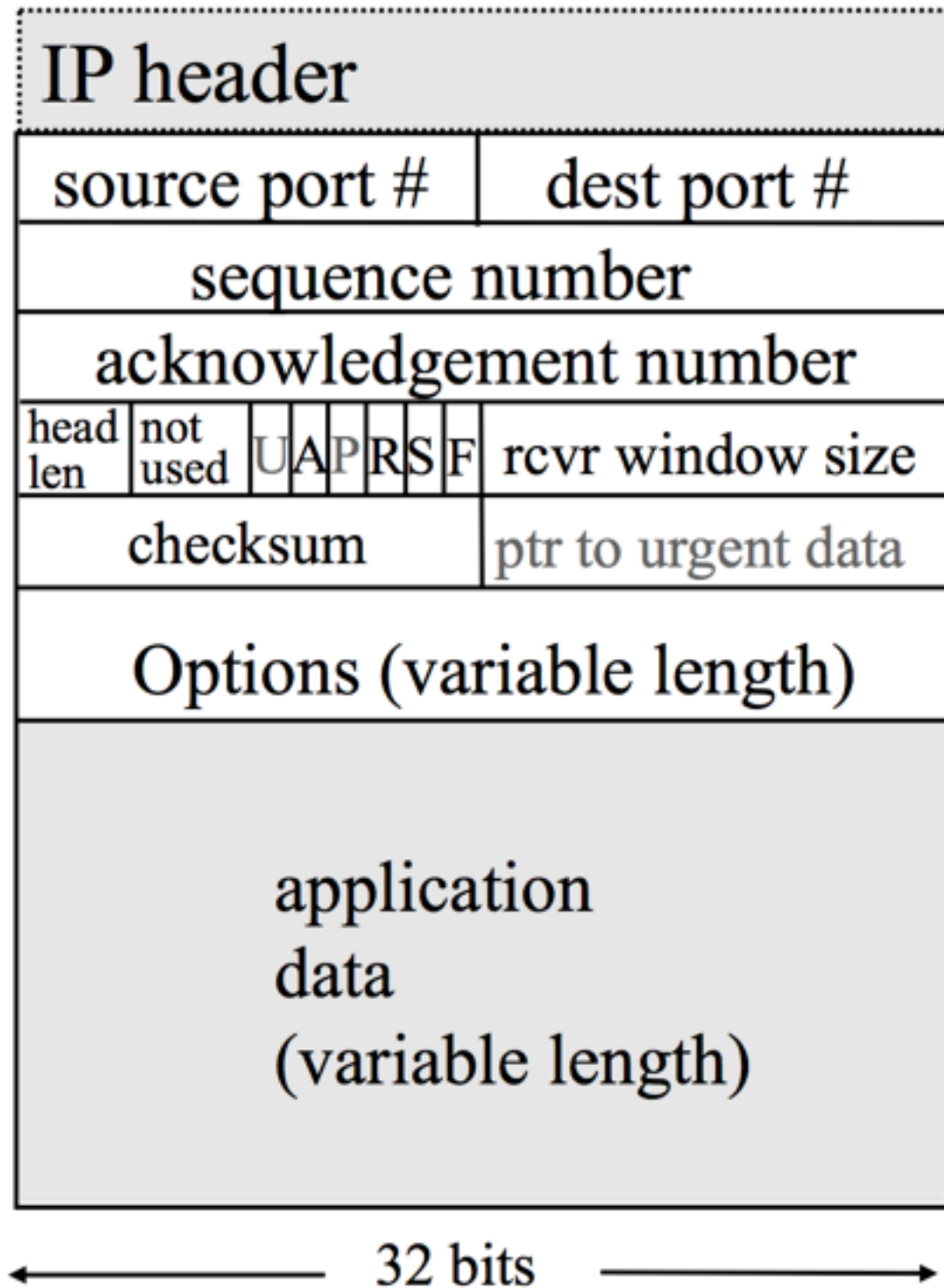


On TCP handshake sequence numbers

- 1) A --> B SYN my sequence number is X
- 2) A <-- B ACK your sequence number is X
- 3) A <-- B SYN my sequence number is Y
- 4) A --> B ACK your sequence number is Y

- Because steps 2 and 3 can be combined in a single message this is called the **three way (or three message) handshake**.
- How X and Y are chosen? Not specified; could be random numbers (using clock), as this is more secure. [RFC 793]
<https://tools.ietf.org/html/rfc793#section-3.3>
<https://support.microsoft.com/en-us/help/172983/>

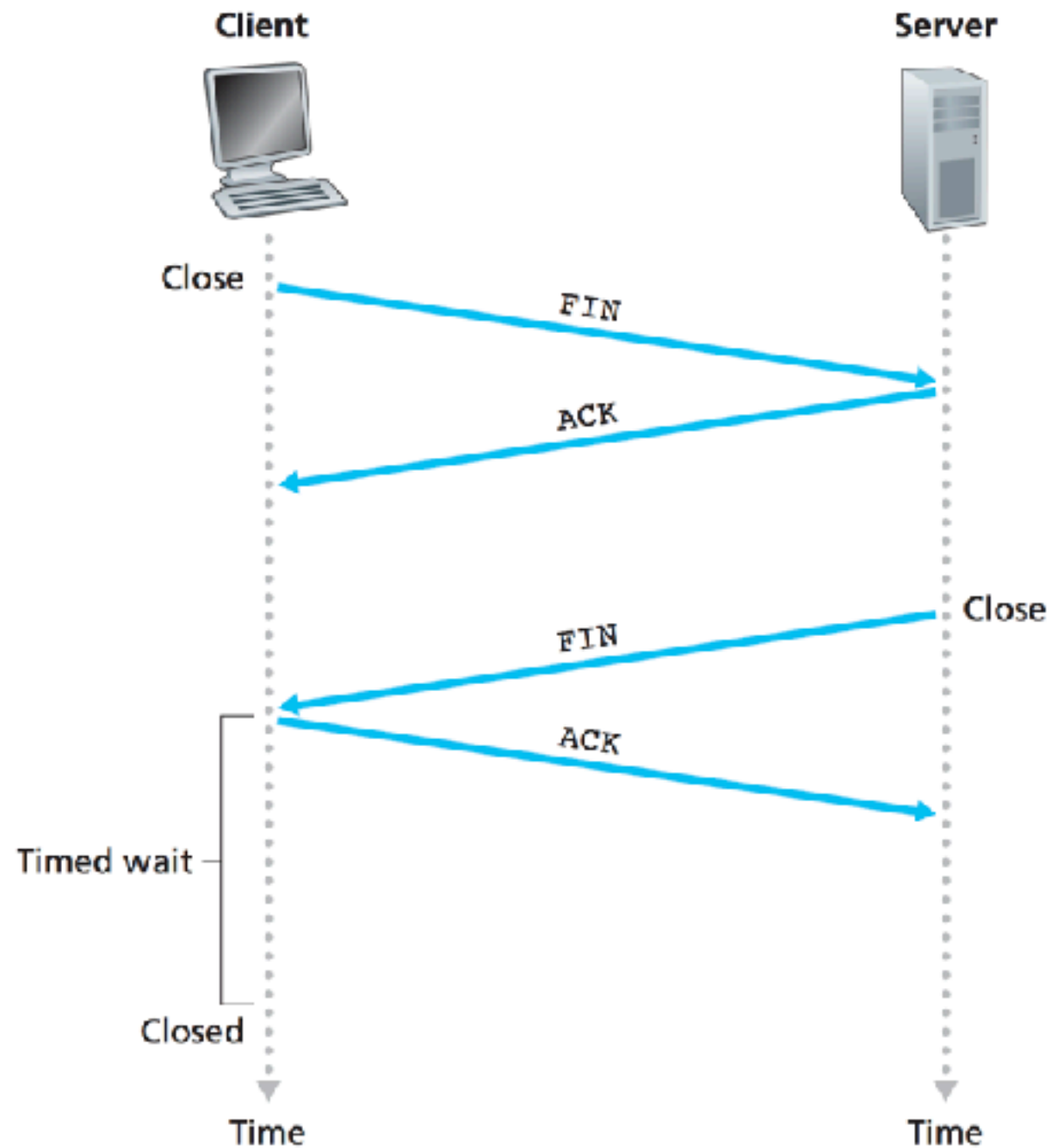
An HTTP 1.0 connection example



TCP: connection teardown

- Normal termination
 - allow unilateral close
 - avoid sequence number overlapping
- TCP must continue to receive data even after closing
 - “Half-open” connection: cannot close connection immediately: what if a new connection restarts and uses same sequence number?

TCP: connection teardown



Two generals'
problem

Figure 3.40 ♦ Closing a TCP connection

TCP: flow control

- Limits the rate a sender transfers data
- Avoid having the sender send data too fast
- Avoid exceeding the capacity of the receiver to process data
- Receiver specify the receive window
- The window size announce the number of bytes still free in the receiver buffer

TCP: timeout

- $\text{EstimatedRTT} = (1 - \alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT}$
- $\text{DevRTT} = (1 - \beta) \times \text{DevRTT} + \beta \times |\text{SampleRTT} - \text{EstimatedRTT}|$
- $\text{Retransmission Timer (RTO)} = \text{EstimatedRTT} + 4 \times \text{DevRTT}$

TCP: timeout

- Karn's algorithm — in case of retransmission
 - do not take the RTT sample (i.e. do not update SRTT or DevRTT)
 - double the retransmission timer value (RTO) after each timeout
 - Take RTT measure again upon next data transmission (that did not get retransmitted)

Examples

References

- [MIT 18.996: Topic in TCS: Internet Research Problems](#)
- [Princeton ELE539A: Optimization of Communication Systems](#)
- <http://www.freesoftware.org/CIE/Course/Section4/>