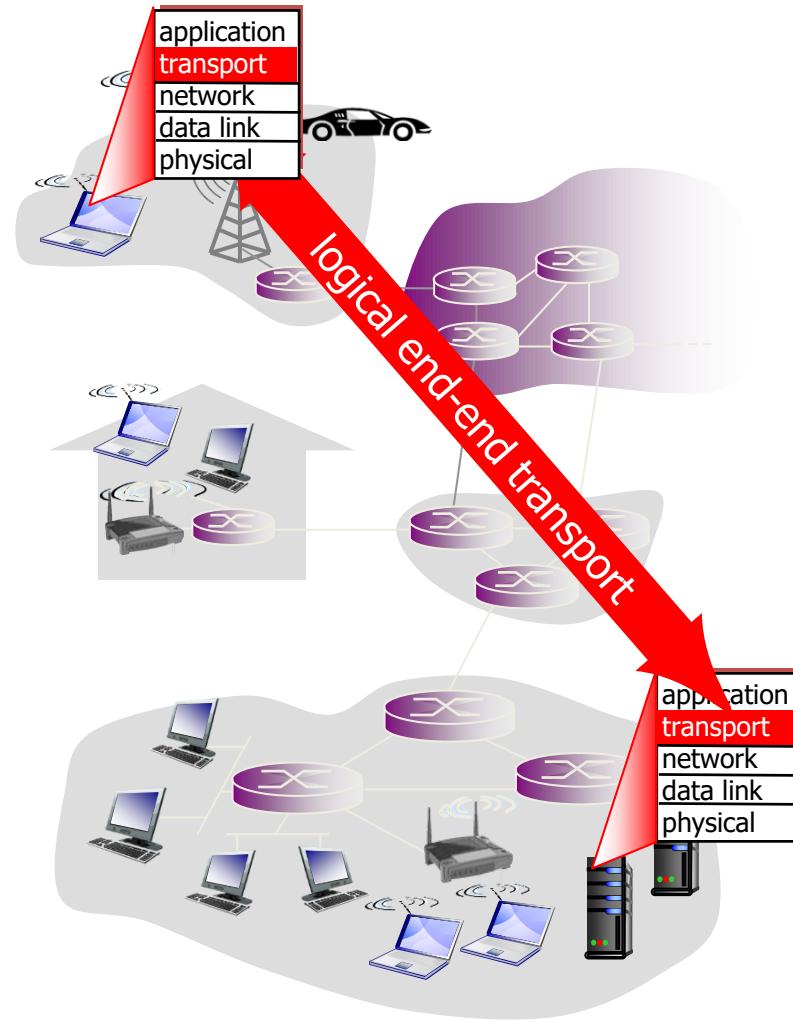


Transport Layer

peterl

Transport level



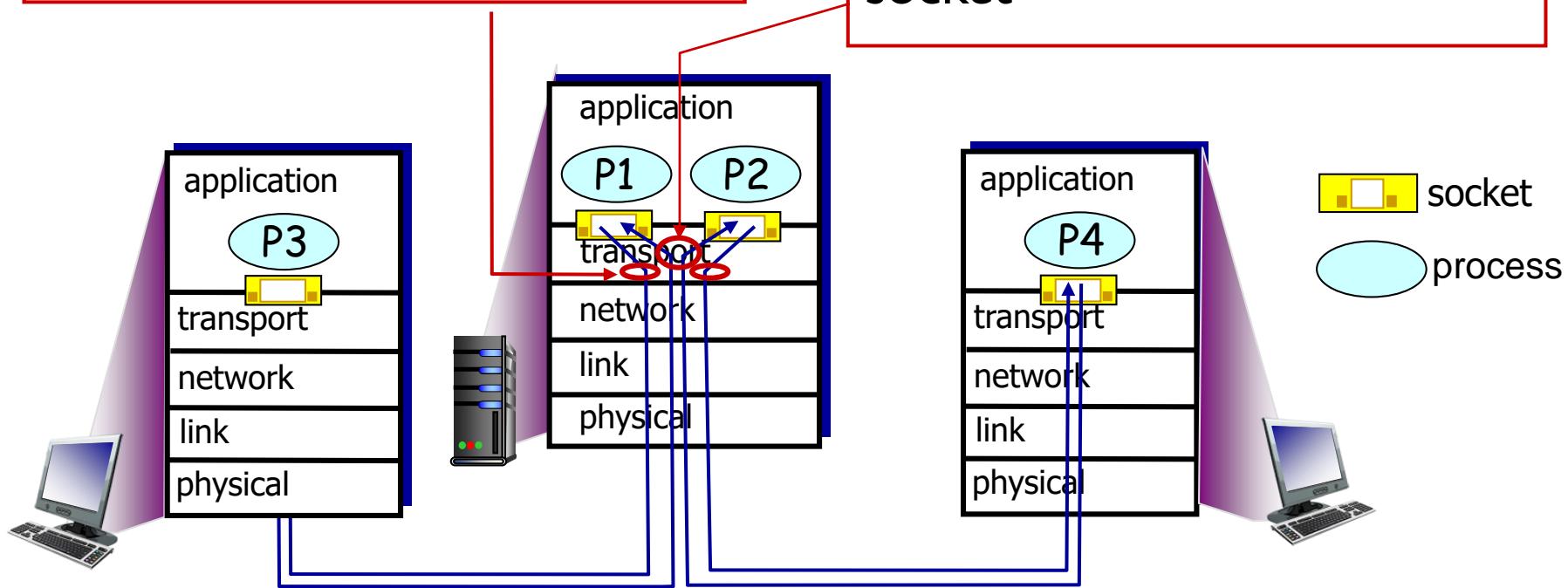
Multiplexing/demultiplexing

- multiplexing at sender:

handle data from multiple sockets, add transport header
(later used for demultiplexing)

- demultiplexing at receiver:

use header info to deliver received segments to correct socket



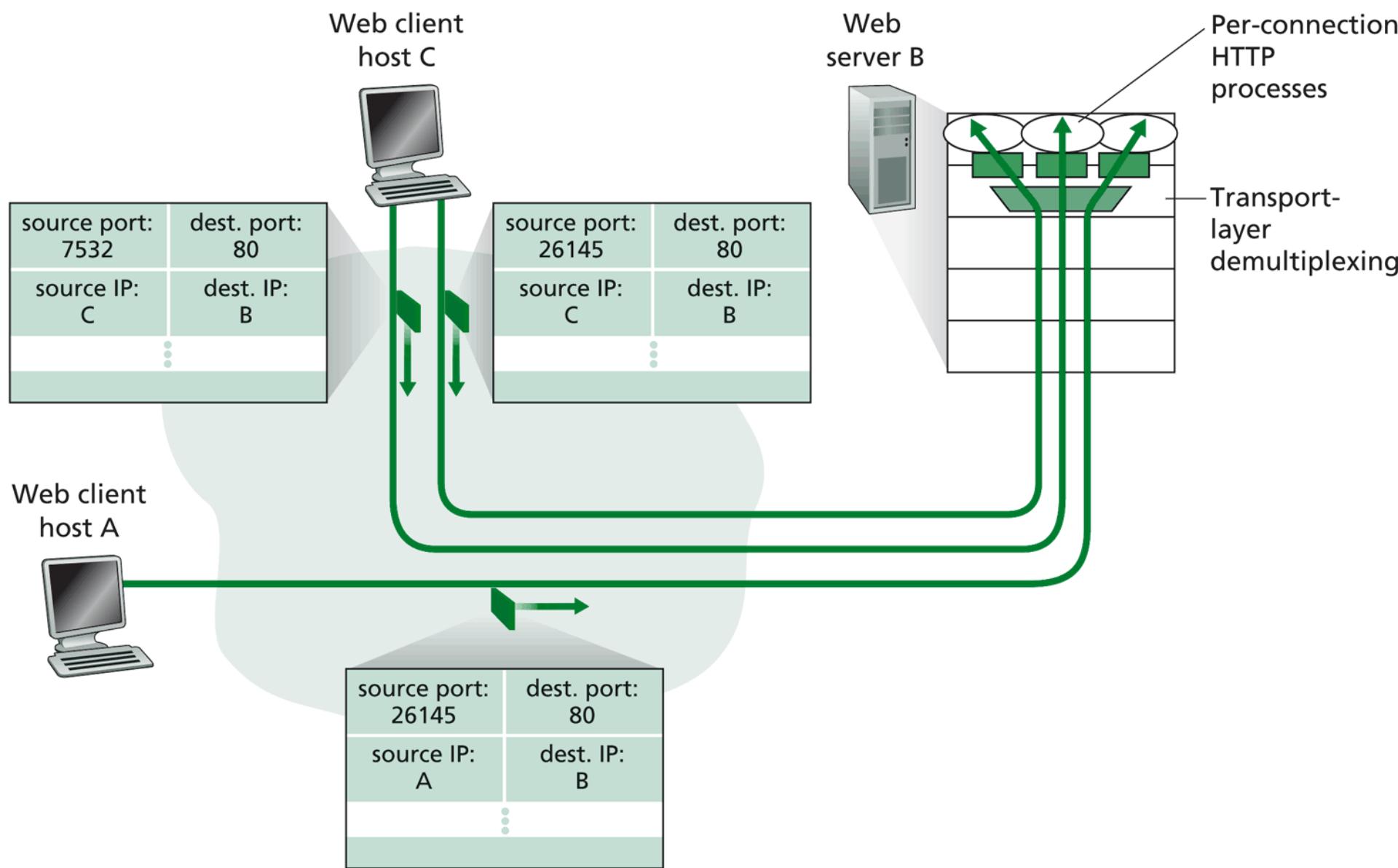
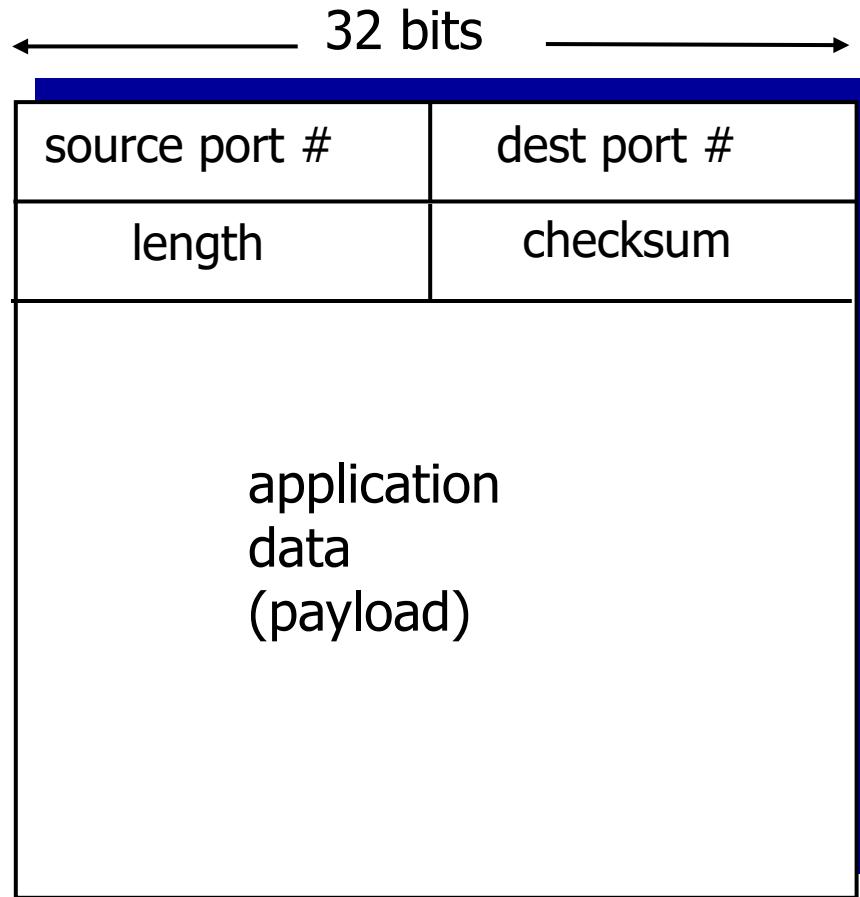


Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

UDP: segment header



UDP segment format

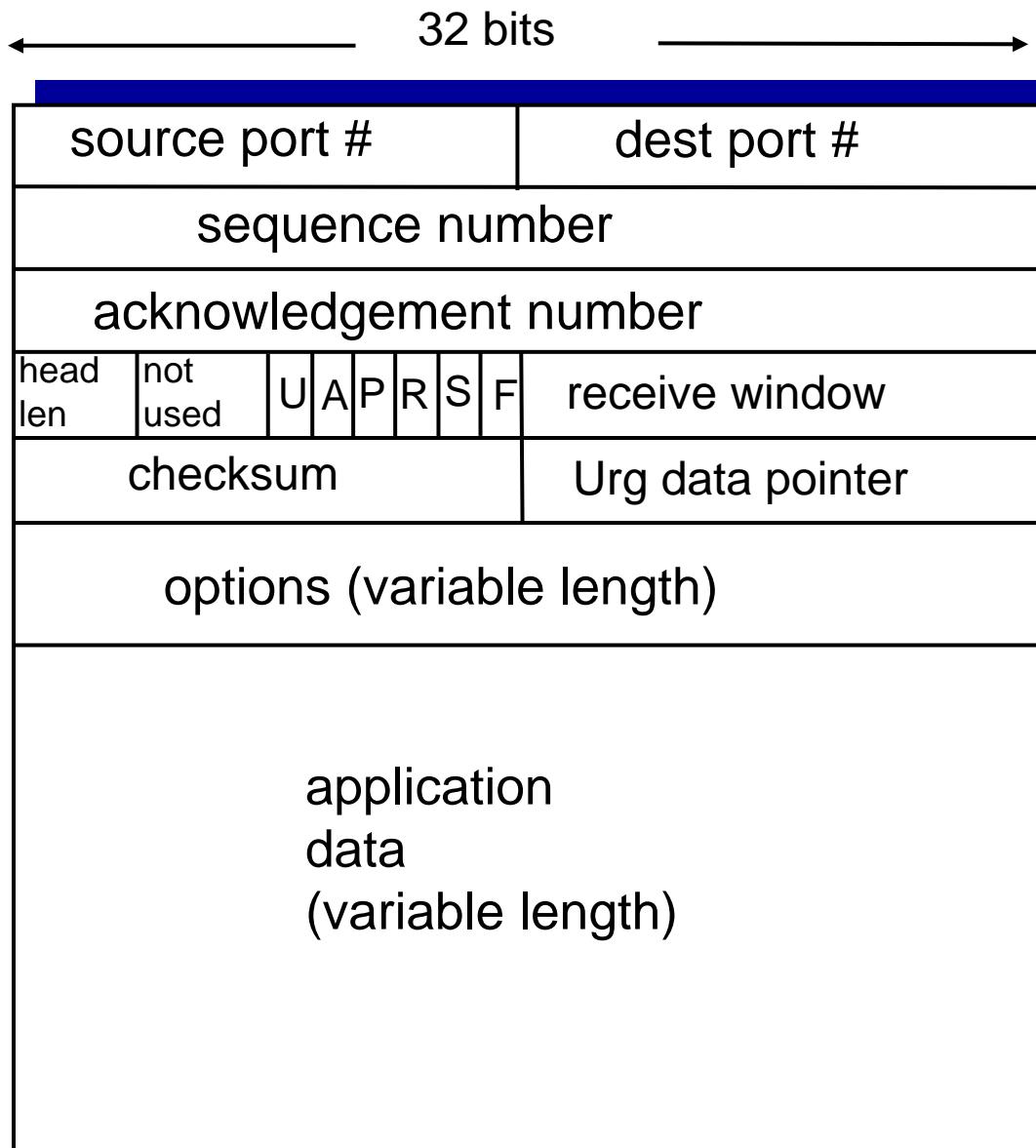
Internet checksum: example

example: add two 16-bit integers

	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
<hr/>																
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

TCP segment structure



TCP 3-way handshake

client state

LISTEN

SYNSENT

choose init seq num, x
send TCP SYN msg



ESTAB

received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

Transport Layer

server state

LISTEN

SYN RCVD



choose init seq num, y
send TCP SYNACK
msg, acking SYN

SYNbit=1, Seq=x

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

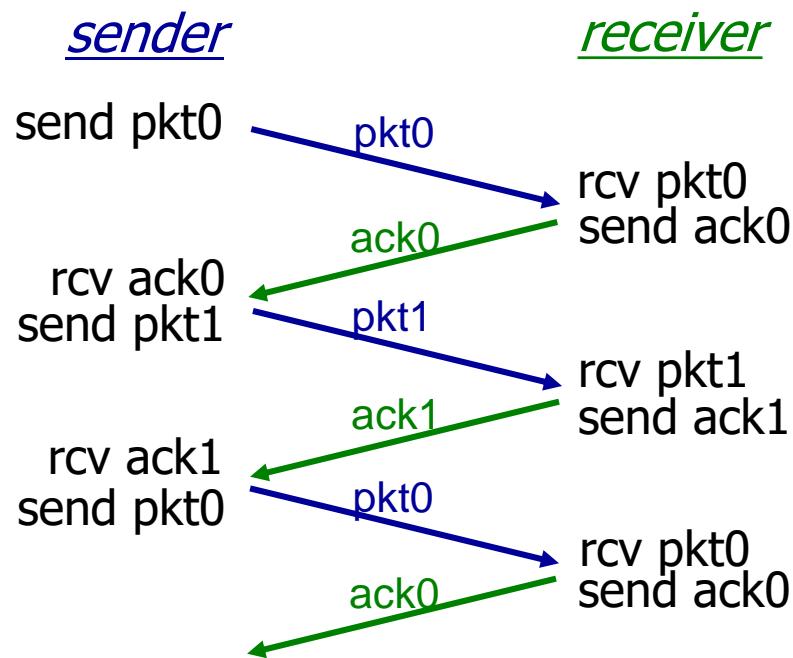
ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

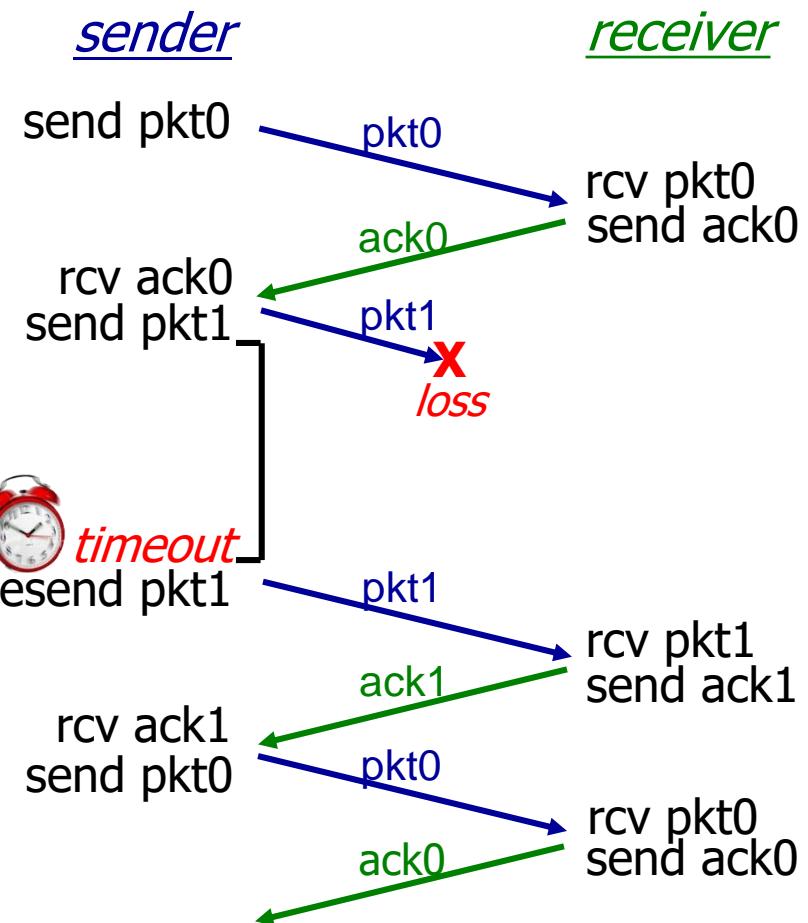
ESTAB

3-8

rdt3.0 in action



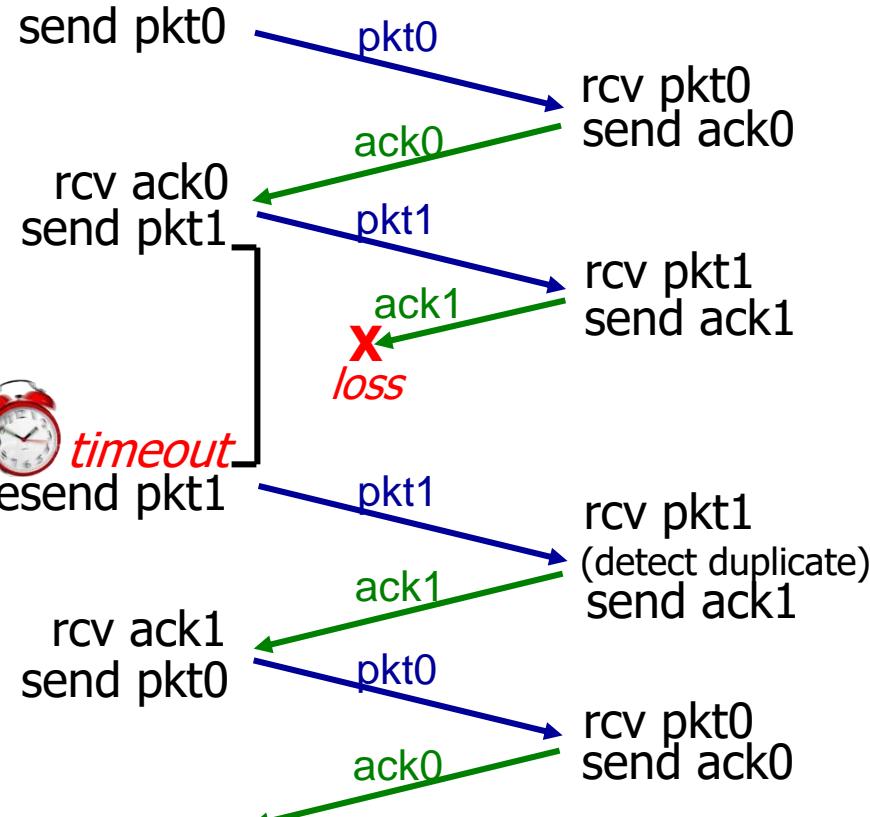
(a) no loss



(b) packet loss

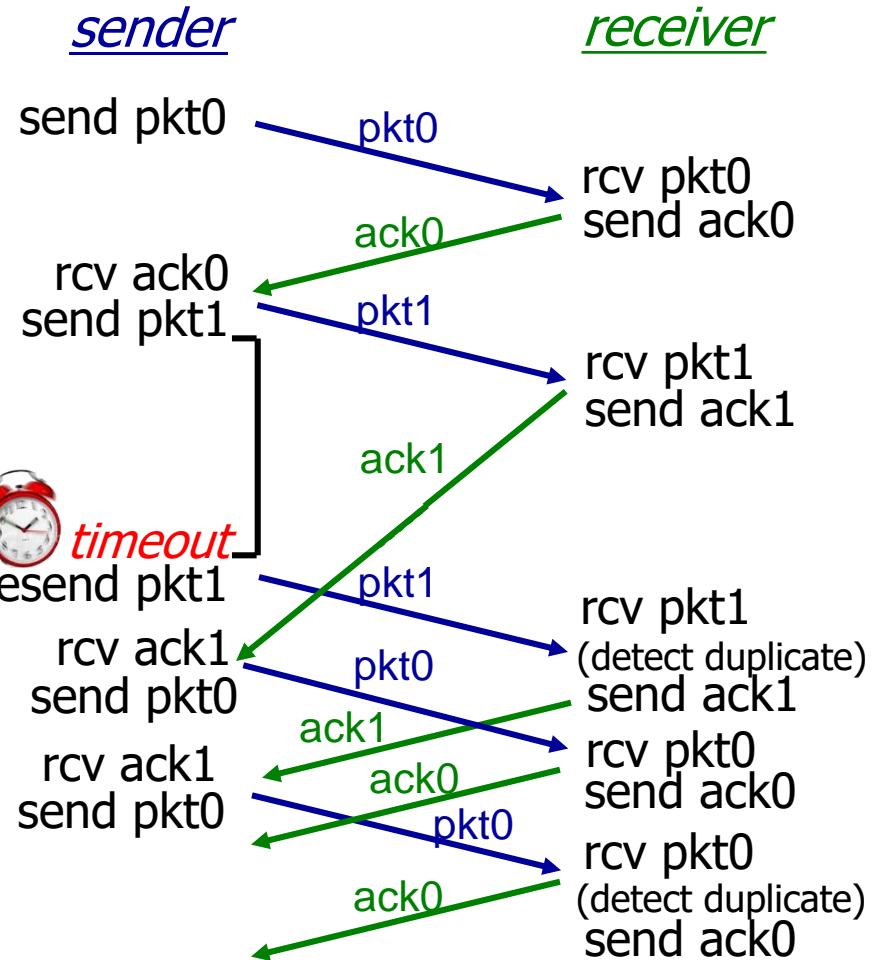
rdt3.0 in action

sender



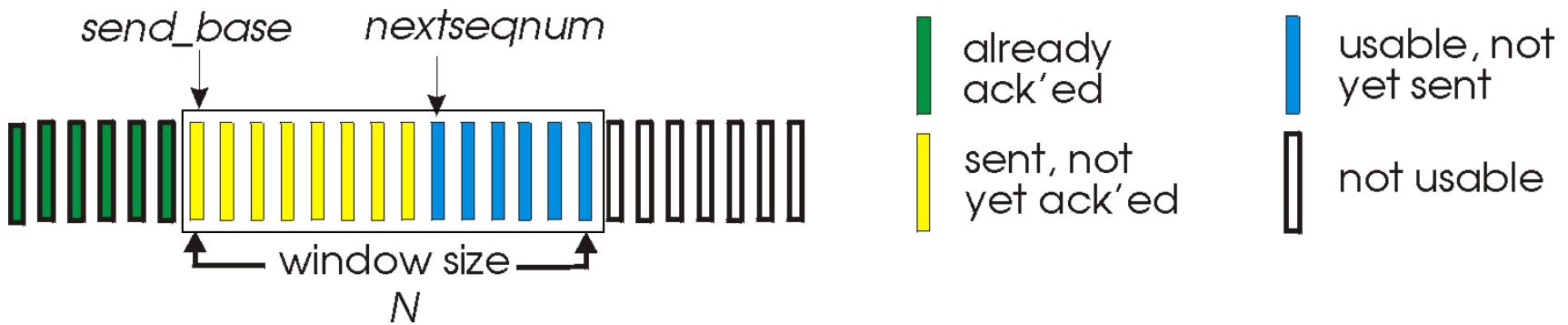
(c) ACK loss

receiver

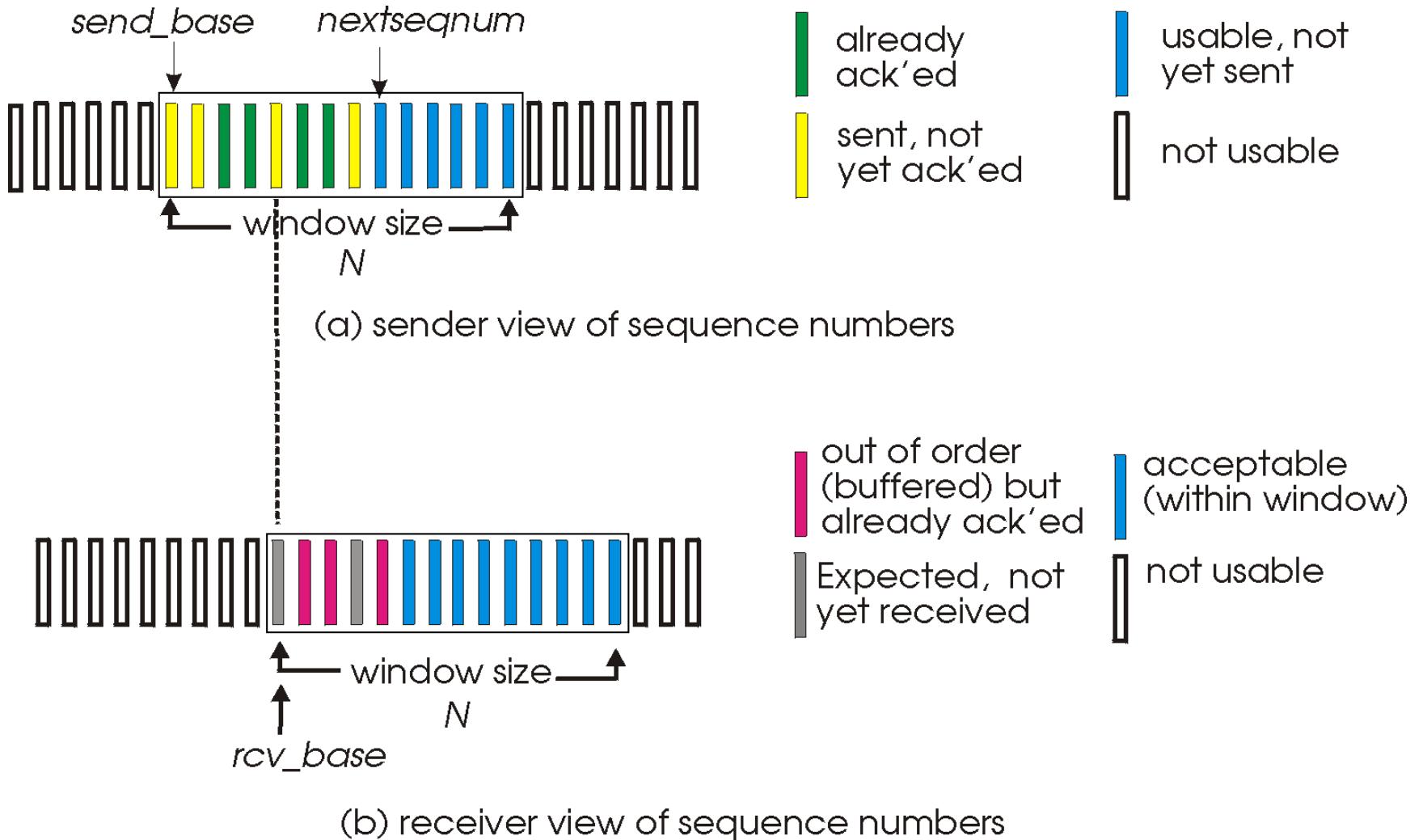


(d) premature timeout/ delayed ACK

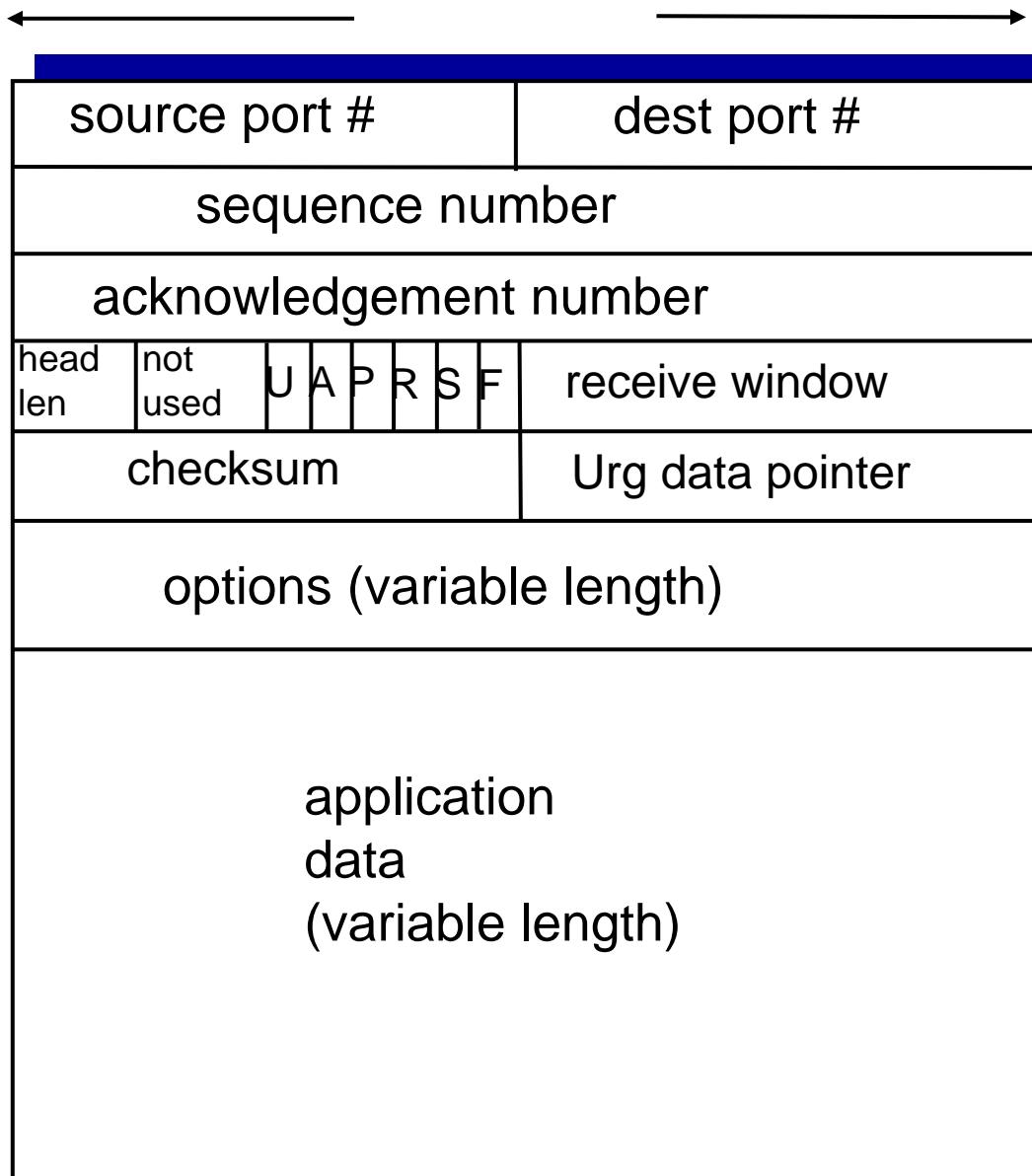
Go Back N



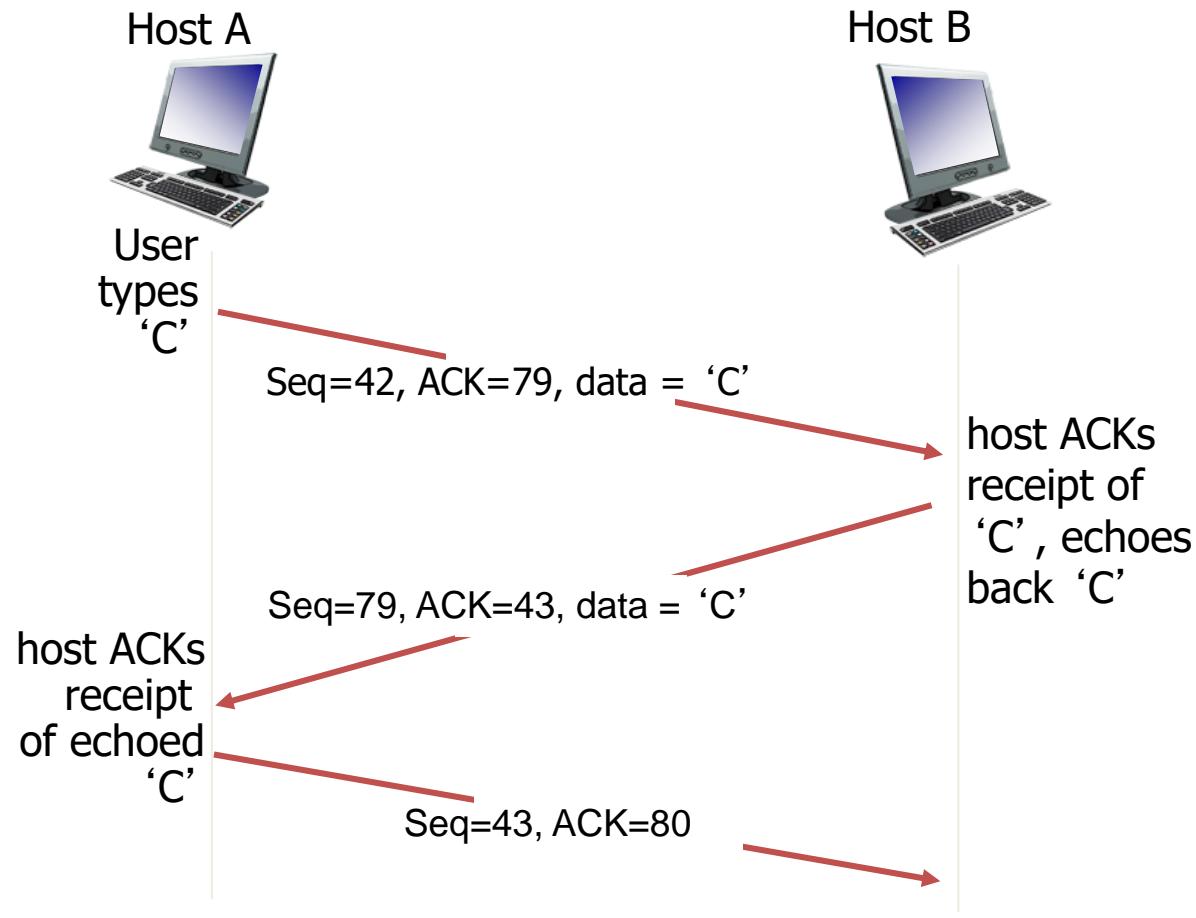
Selective Repeat



TCP segment structure



TCP seq. numbers, ACKs



simple telnet scenario

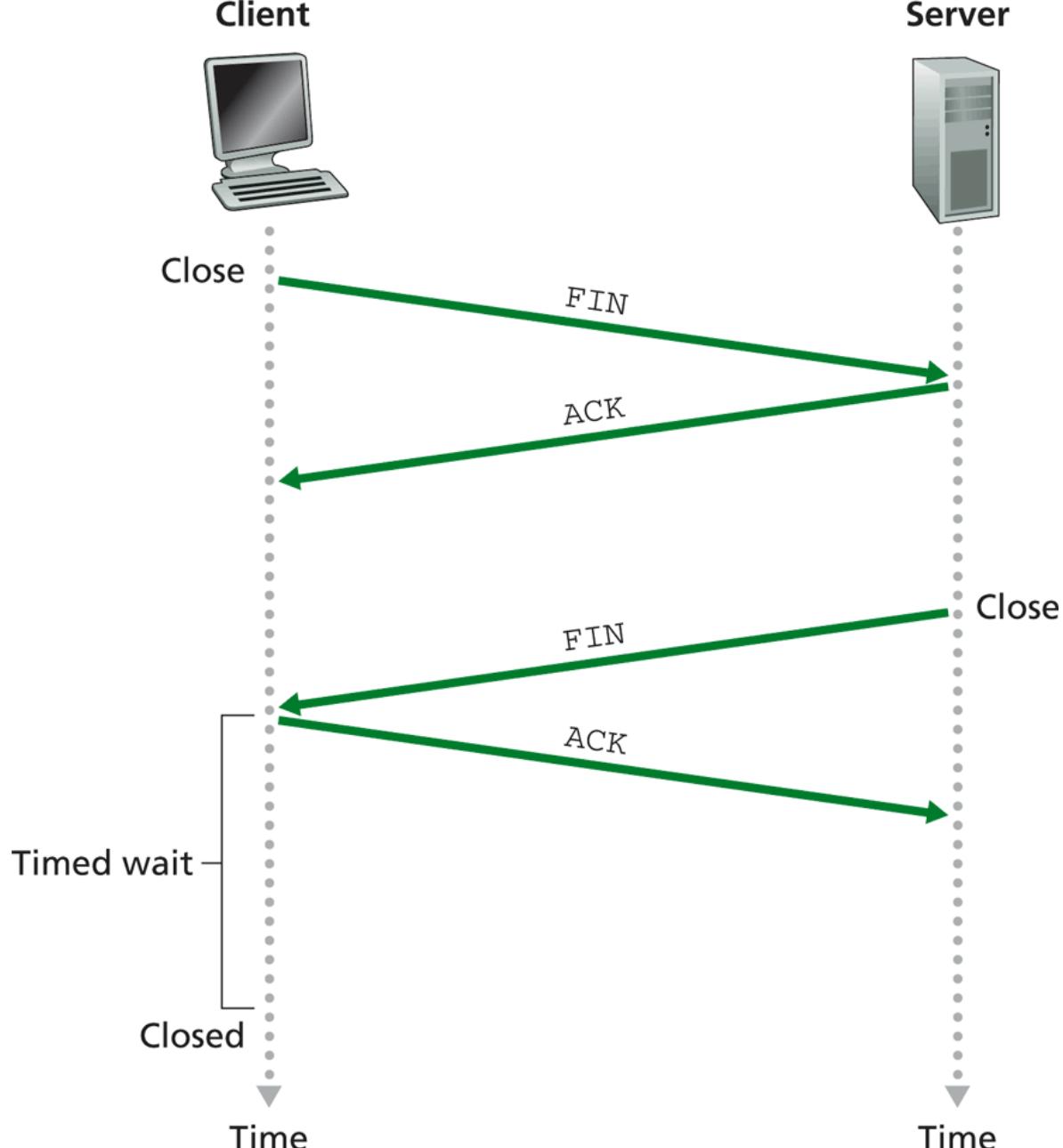


Figure 3.39 ♦ Closing a TCP connection