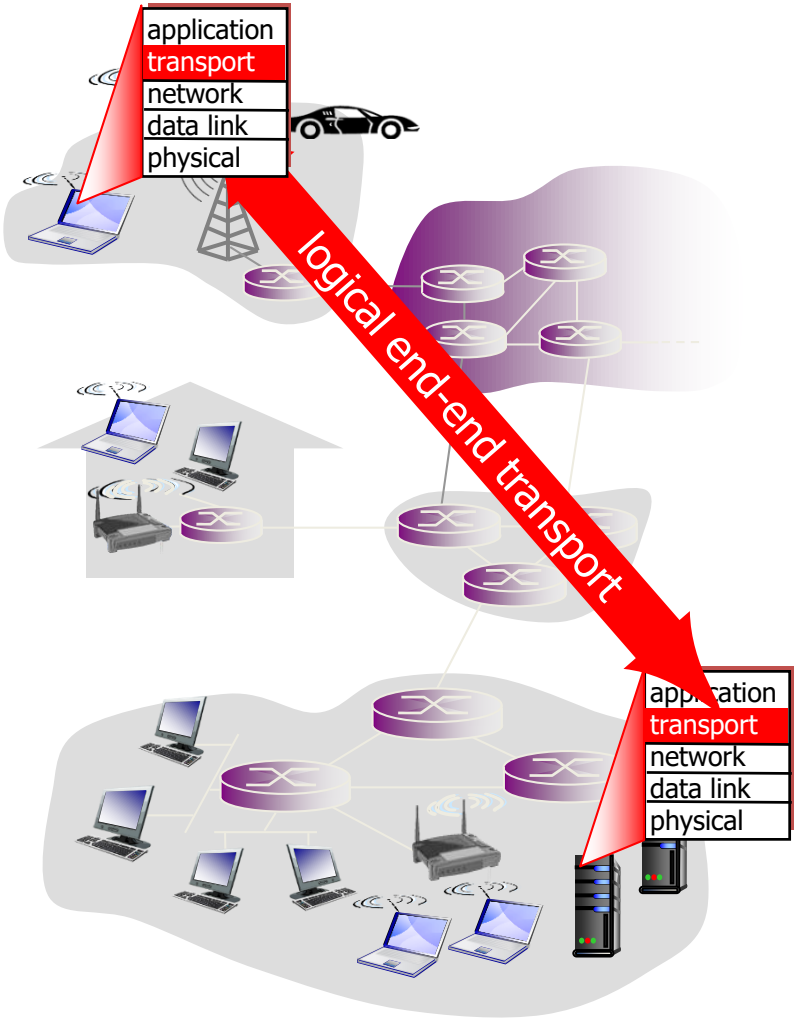
Transport Layer

peterl

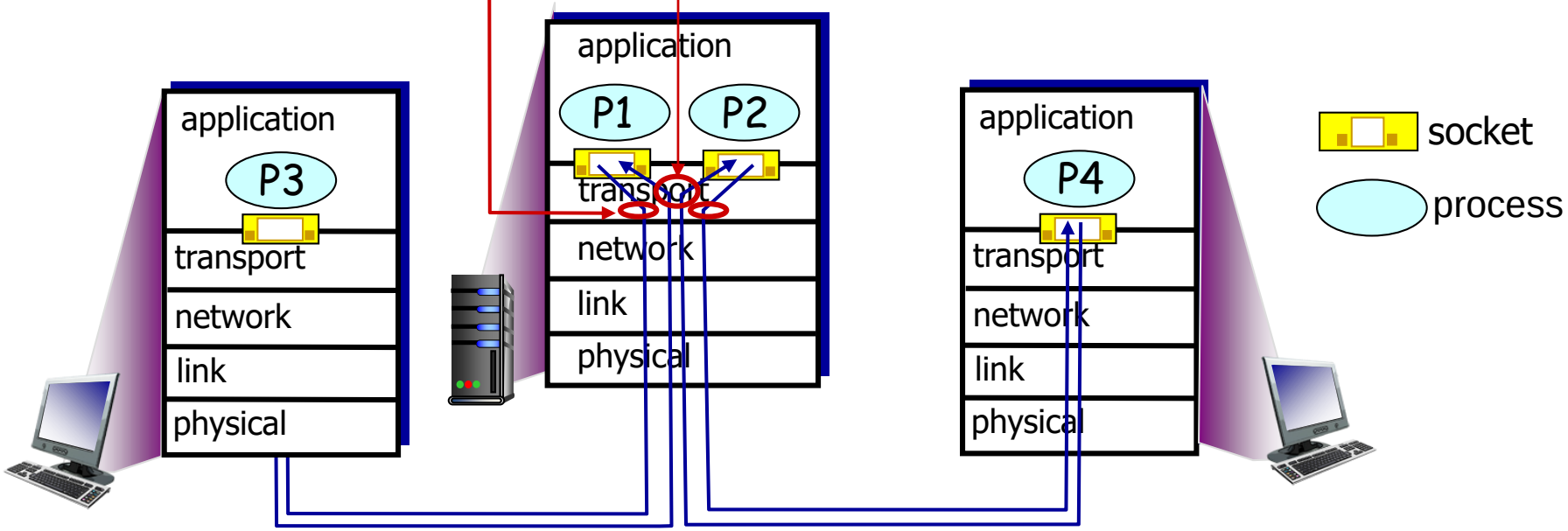
Transport level



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



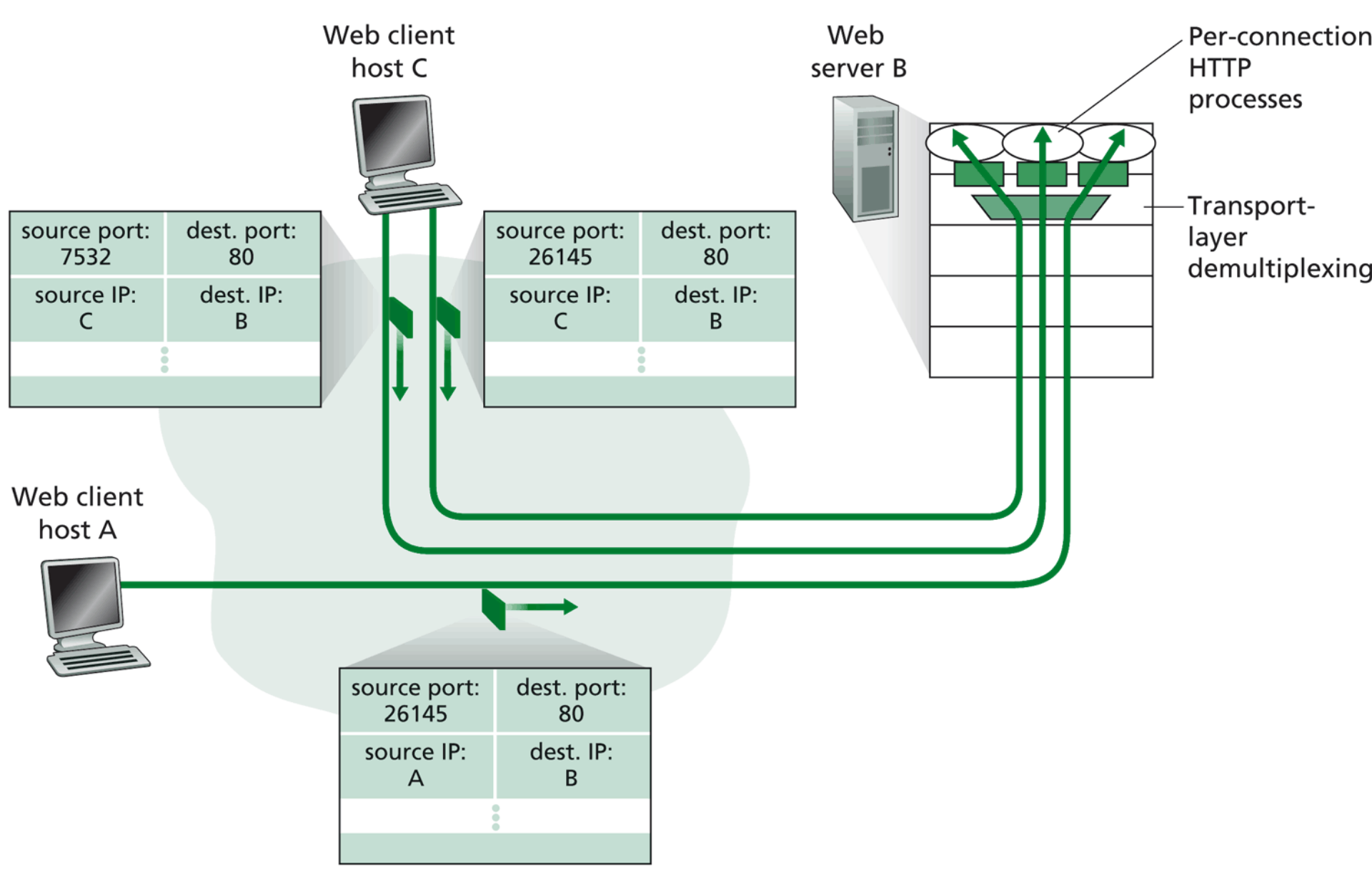
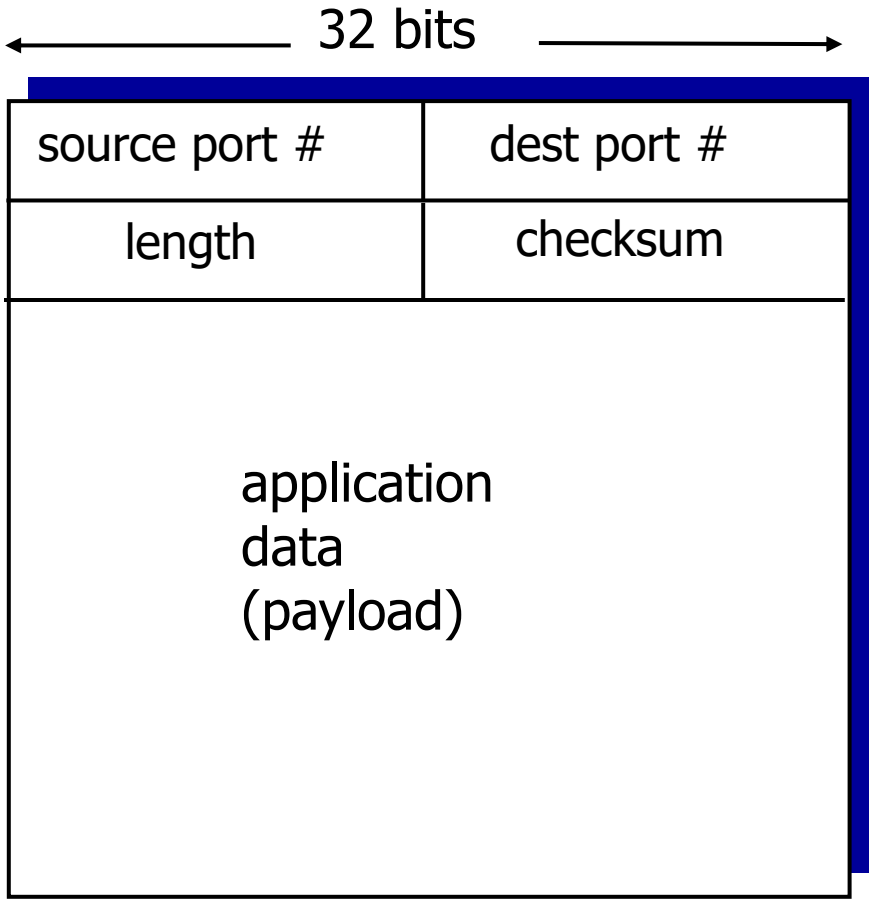


Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

UDP: segment header



UDP segment format

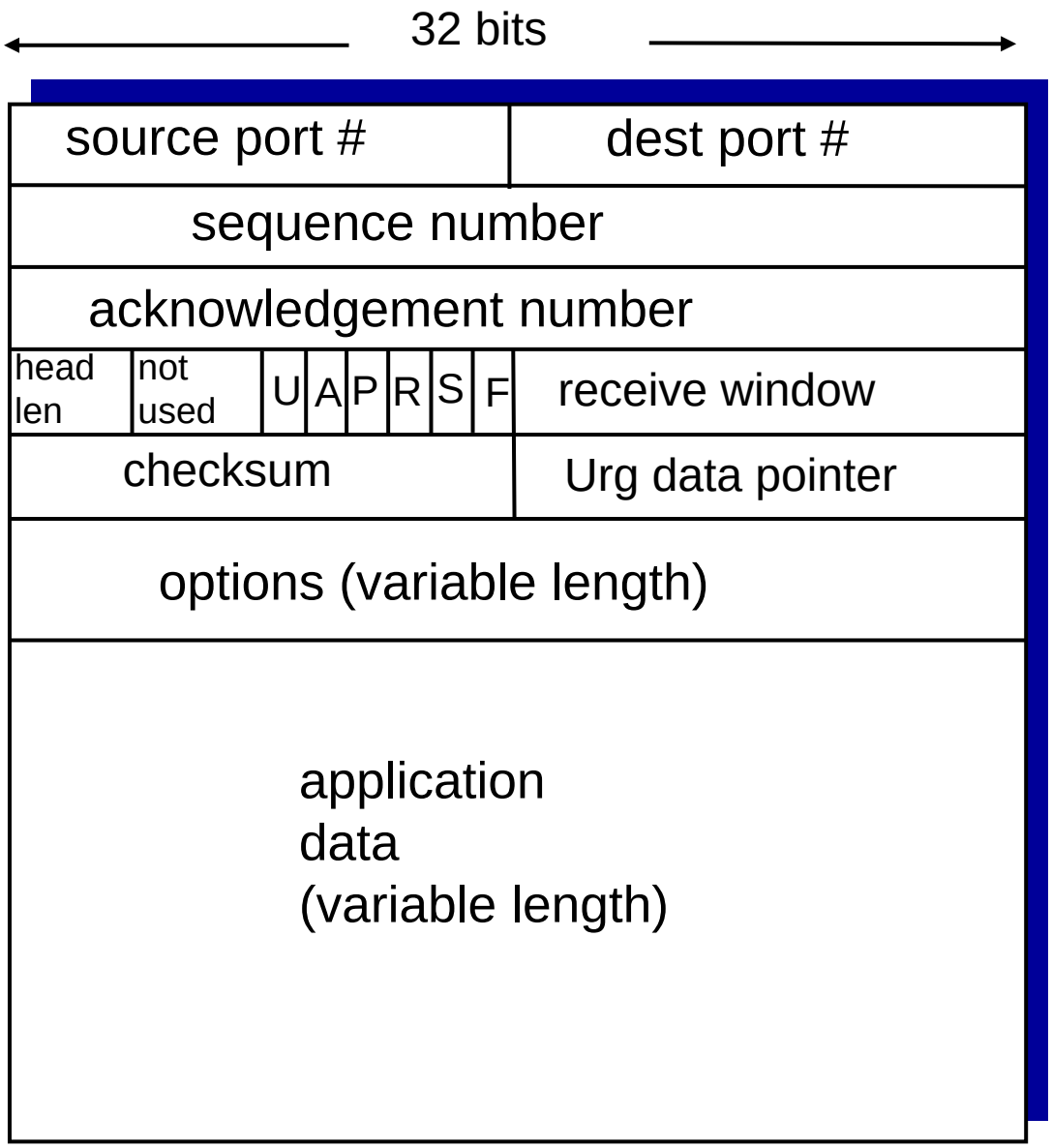
Internet checksum: example

example: add two 16-bit integers

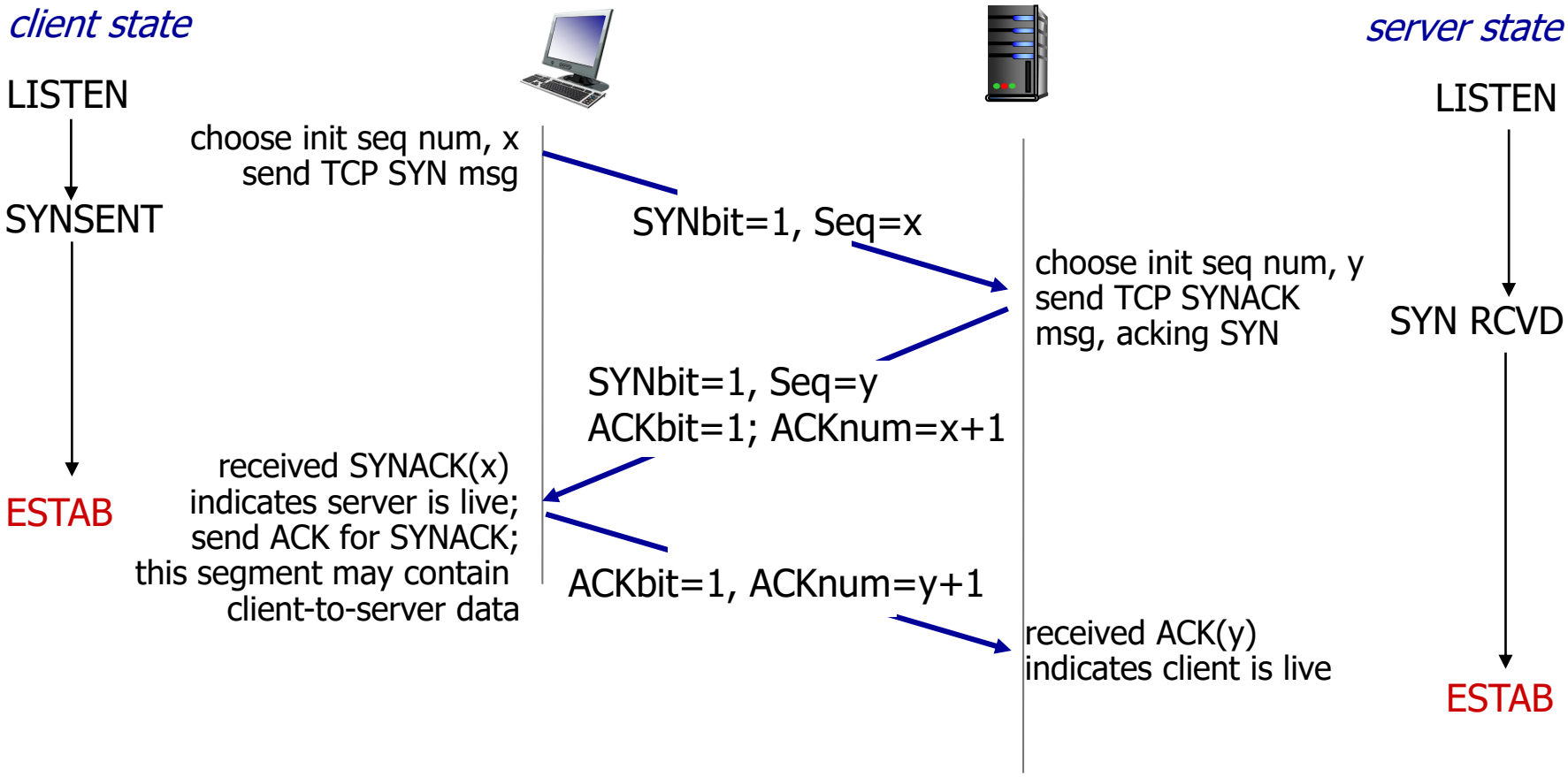
	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

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

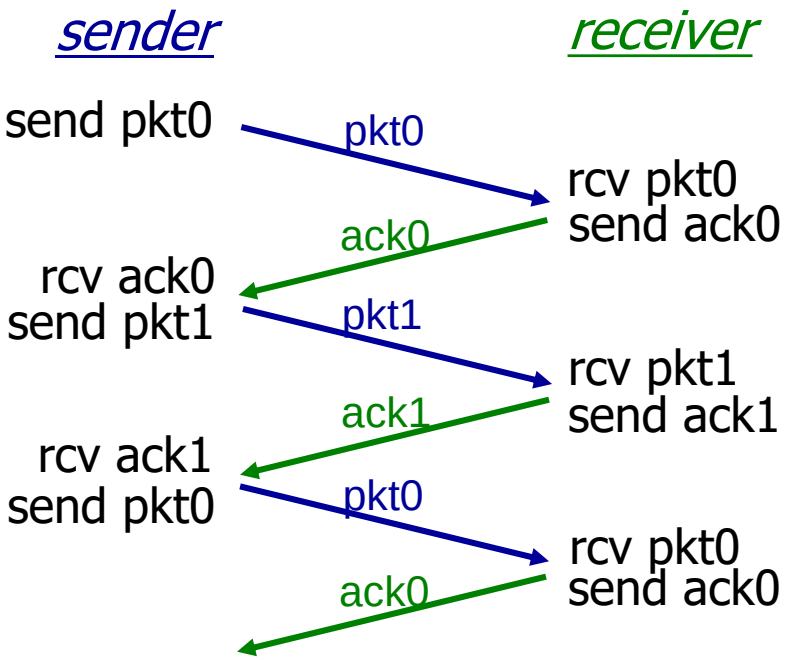
TCP segment structure



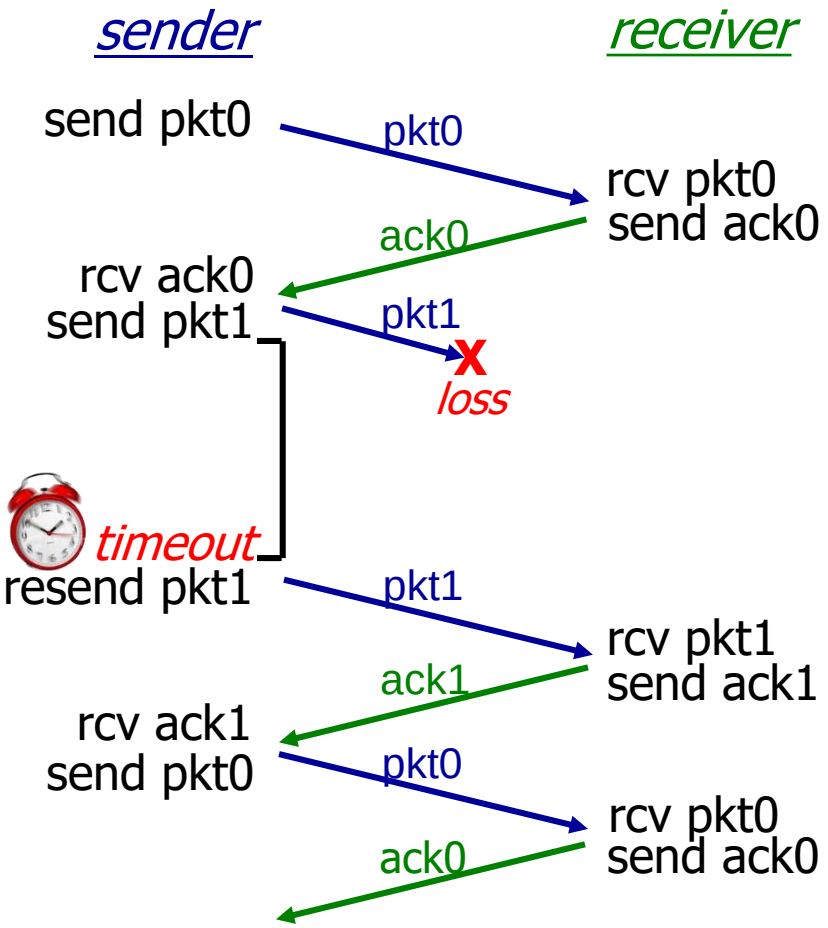
TCP 3-way handshake



rdt3.0 in action

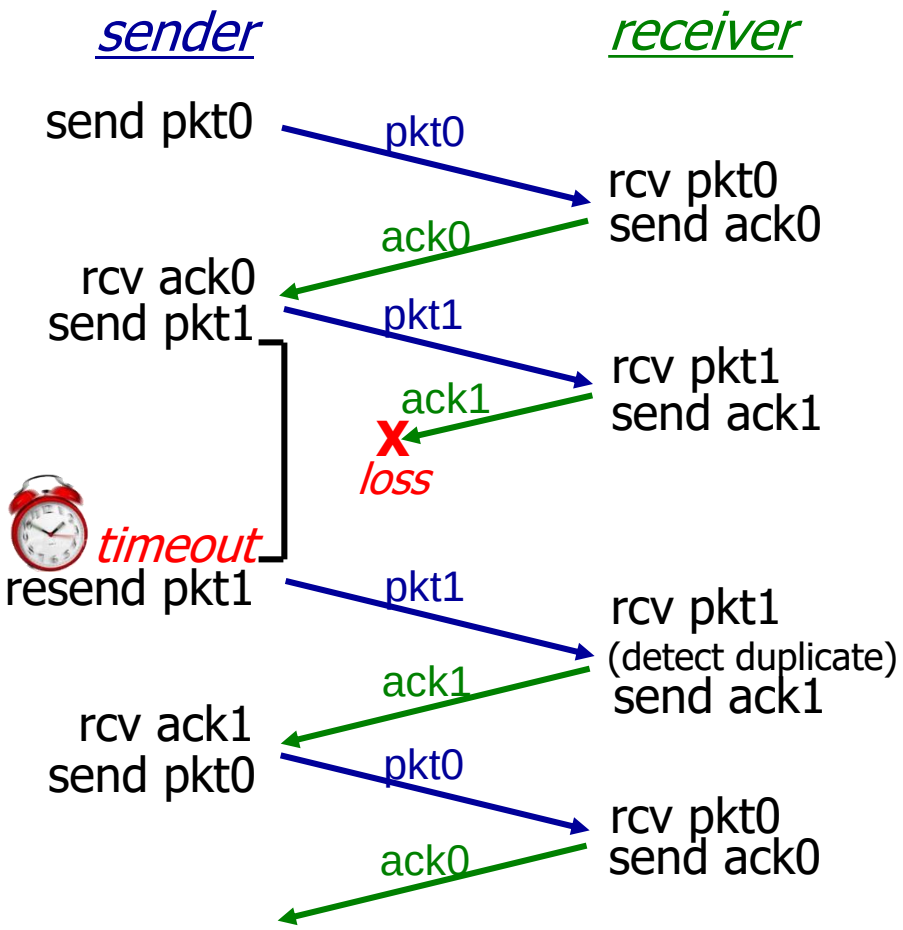


(a) no loss

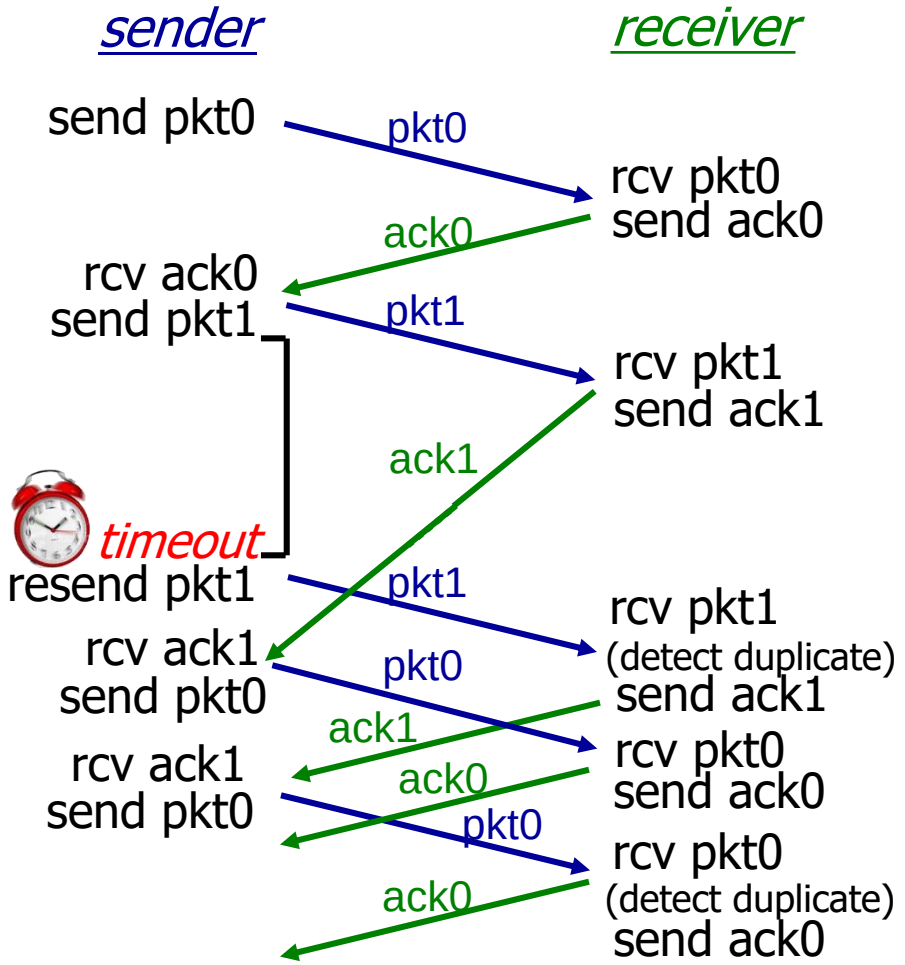


(b) packet loss

rdt3.0 in action

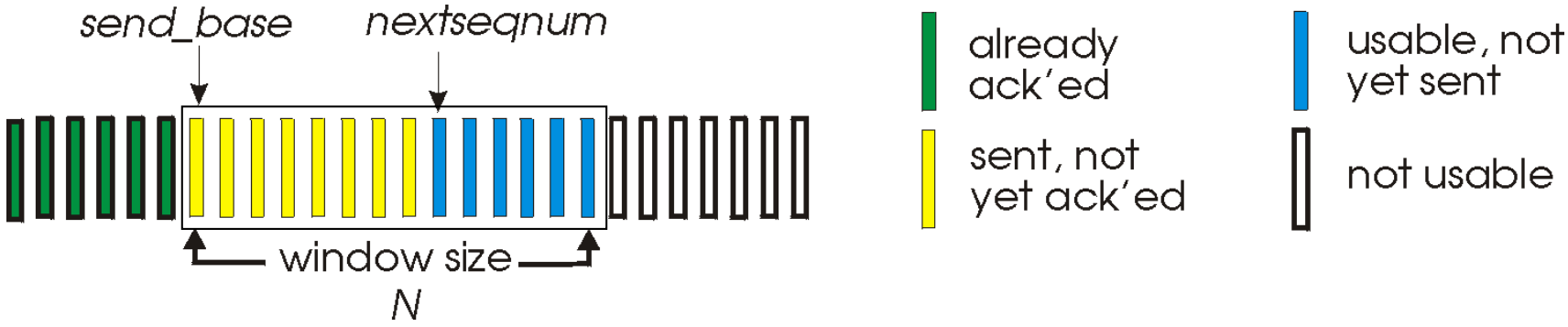


(c) ACK loss

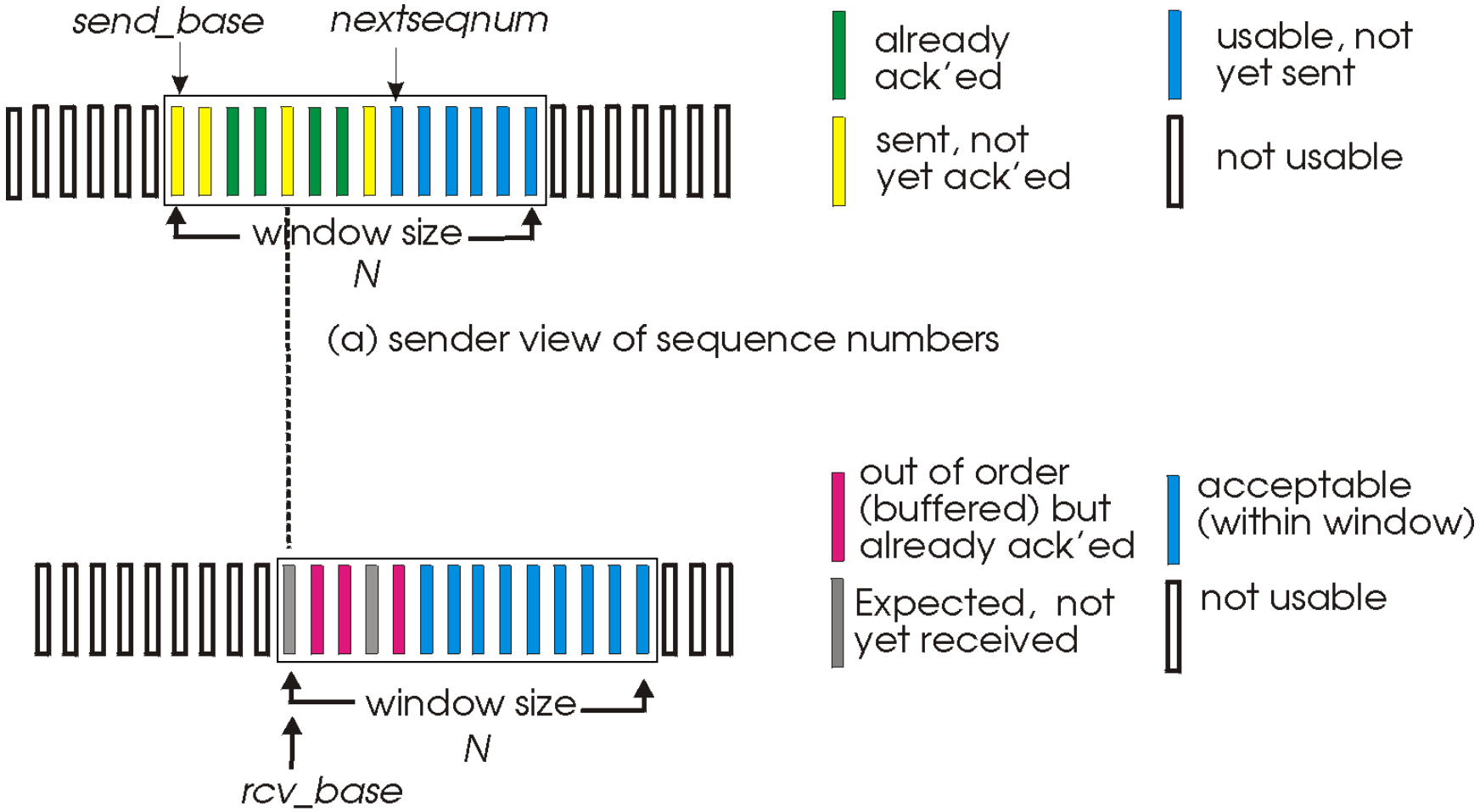


(d) premature timeout/ delayed ACK

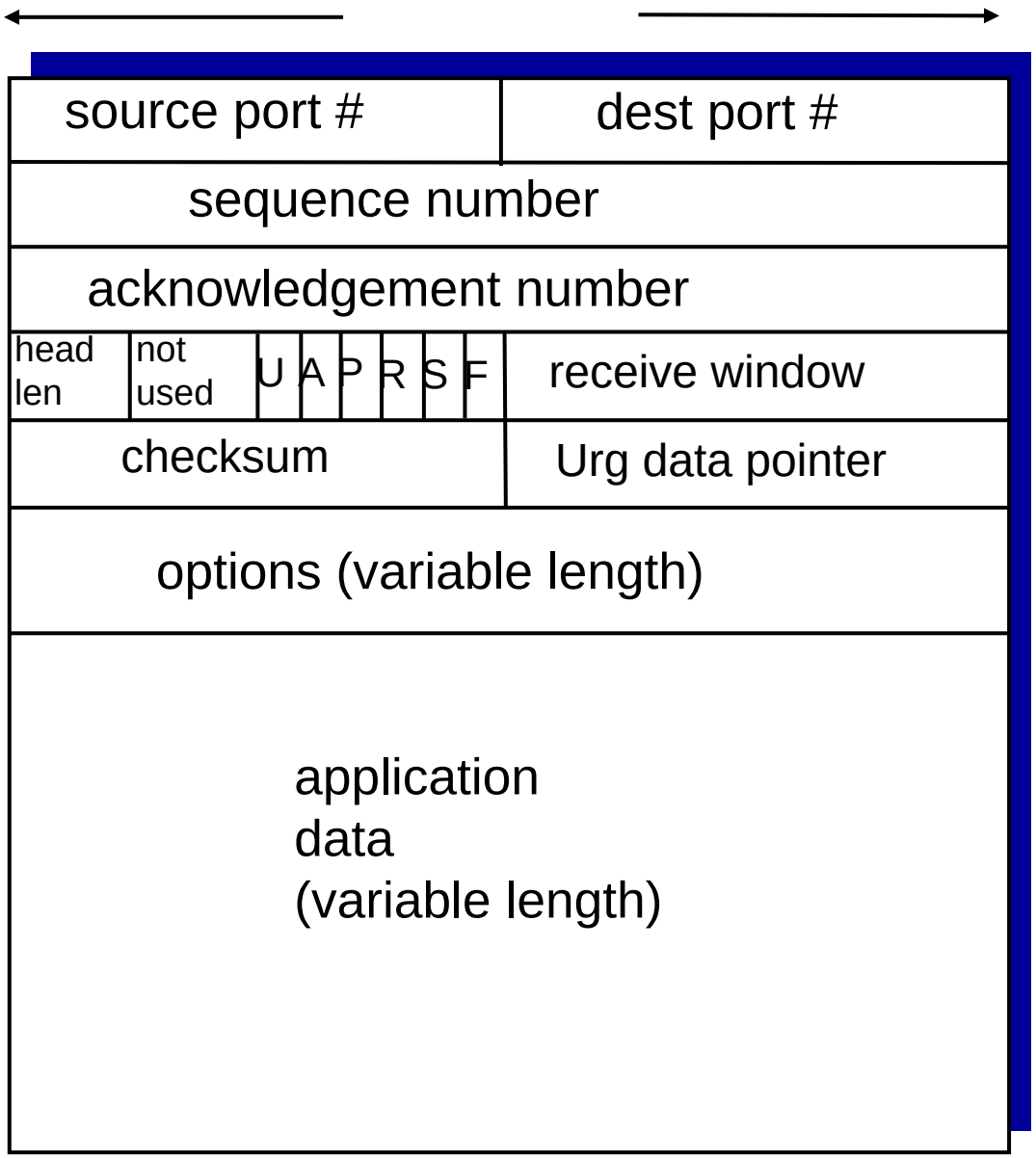
Go Back N


send_base
nextseqnum
window size
N
already ack'ed
sent, not yet ack'ed
usable, not yet sent
not usable

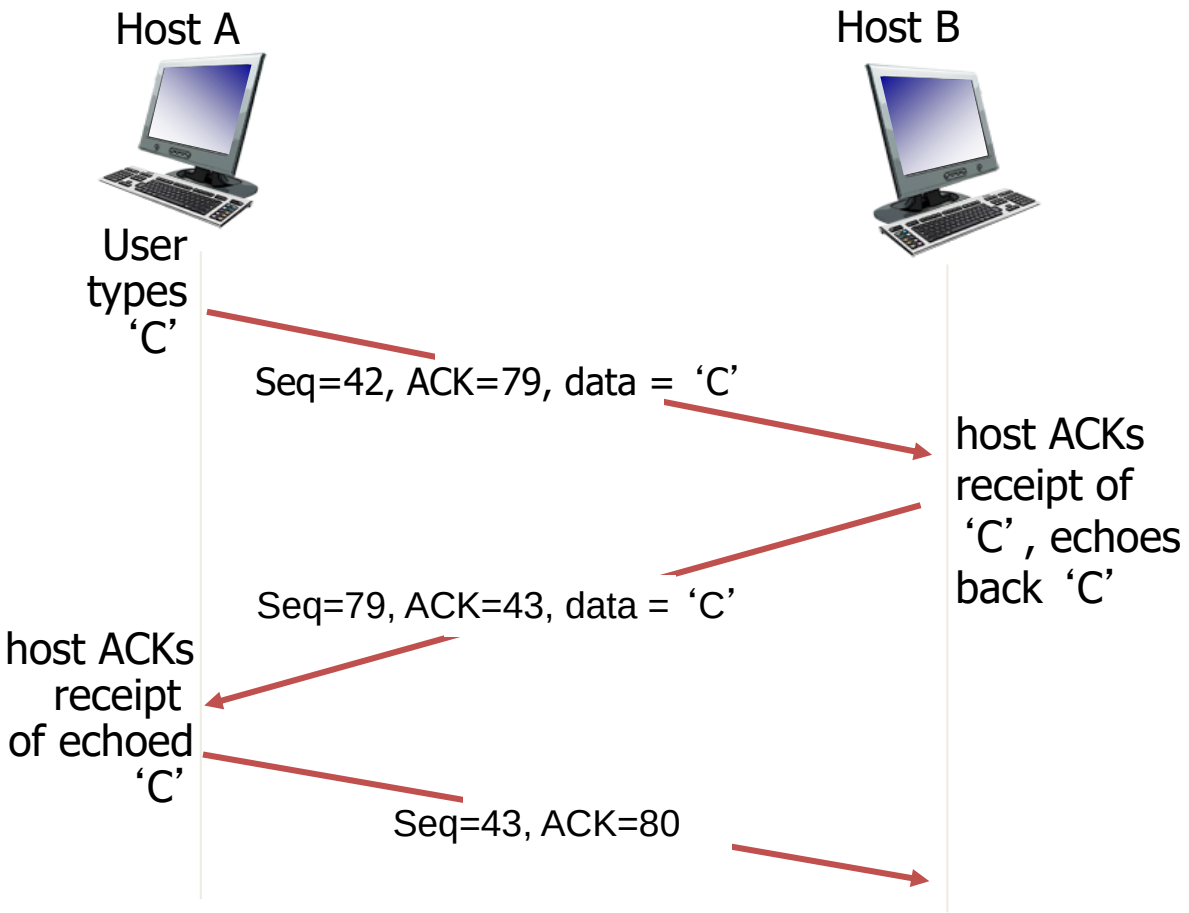
Selective Repeat



TCP segment structure



TCP seq. numbers, ACKs



simple telnet scenario

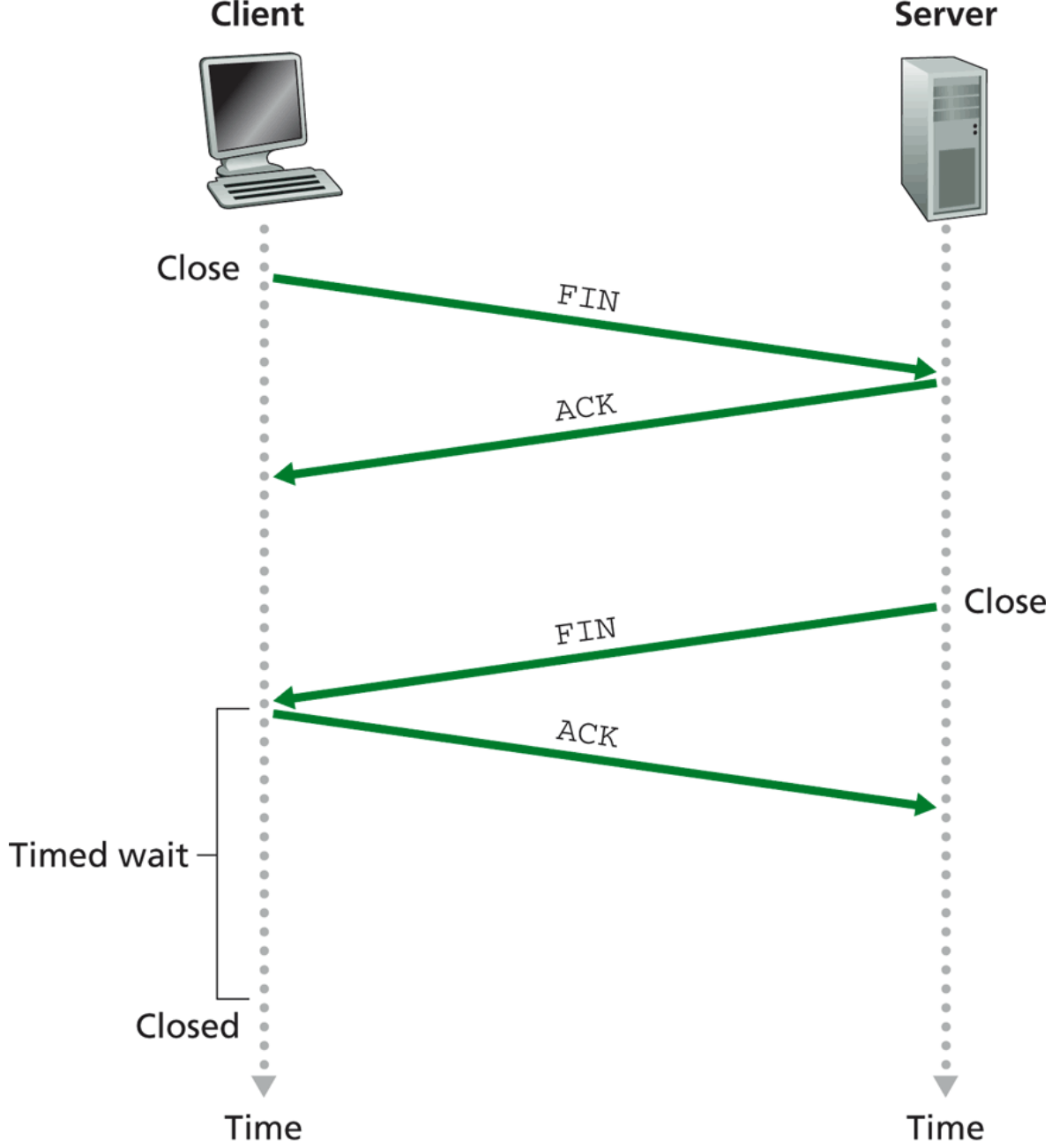


Figure 3.39 ♦ Closing a TCP connection