Which of the following are all layers of the Internet Protocol Stack?

a. Link, Presentation, Application, Transport

[✓ b. Physical, Transport, Application, Network]

c. Application, Transport, Network, Presentation

d. Transport, Network, Link, Session

e. None of the above

What part(s) of a router would typically be implemented in hardware and not software?

a. Input Ports

b. Output Ports

c. Switching Fabric

[✓ d. All of the Above]

e. None of the above

Consider sending host H1 and receiving host H2 with a single packet switch between them. The transmission rates between the sending host and the receiving host are R1 and R2, respectively. Assuming that the switch uses store-and-forward packet switching, what is the total end-to-end delay to send a packet of length L? (only consider transmission delay in this scenario)

a. R1/L + R2/L

b. (2R1 + 2R2)/L

c. L/(2R1+2R2)

[✓ d. L/R1 + L/R2]

e. (R1 + R2)/L

DNS typically runs on the following transport protocol(s):
a. TCP
[✓ b. UDP]
c. HTTP
d. CNAME
e. All of the above

DNS needs to be lightweight, a connection based protocol would add overhead.

4 hosts are communicating with a server via UDP port 5000. How many sockets in total will have been created on the server? What if the hosts were communicating via TCP connections?

a. UDP: 4 sockets; TCP: 4 sockets

b. UDP: 1 socket; TCP: 4 sockets

<u>c. UDP: 1 socket; TCP: 5 sockets</u>

d. UDP: 4 sockets; TCP: 1 socket

e. UDP: 5 sockets; TCP: 1 socket

Don’t forget the welcoming socket for TCP connections! 4 sockets for TCP, one welcoming socket. UDP is connectionless.

In class, we defined traffic intensity mathematically as La/R, where L is the average packet length, a is the average arrival rate, and R is the transmission rate of the outbound link. Packet delays become significant

a. as La/R approaches zero

b. as La approaches one

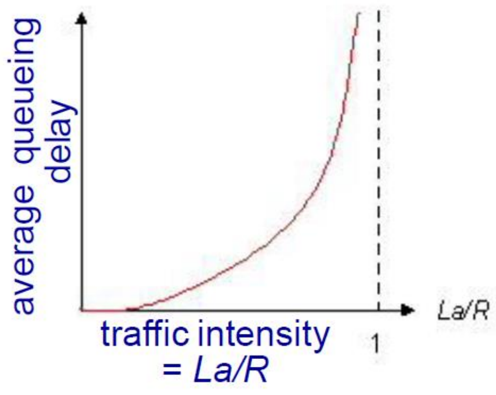
c. as R approaches one

[✓ d. as La approaches R]

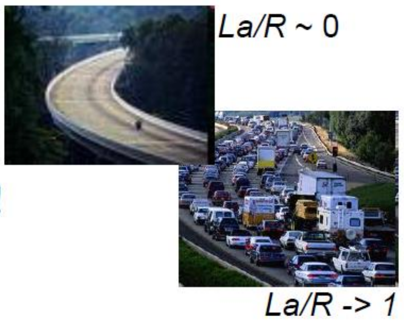
e. None of the above

Queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- a: average packet arrival rate



- *La/R* ~ 0: avg. queueing delay small
- *La/R* ≤ 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving than can be serviced, average delay infinite!



Introduction 1-48

UDP provides:

a. A reliable data transfer service

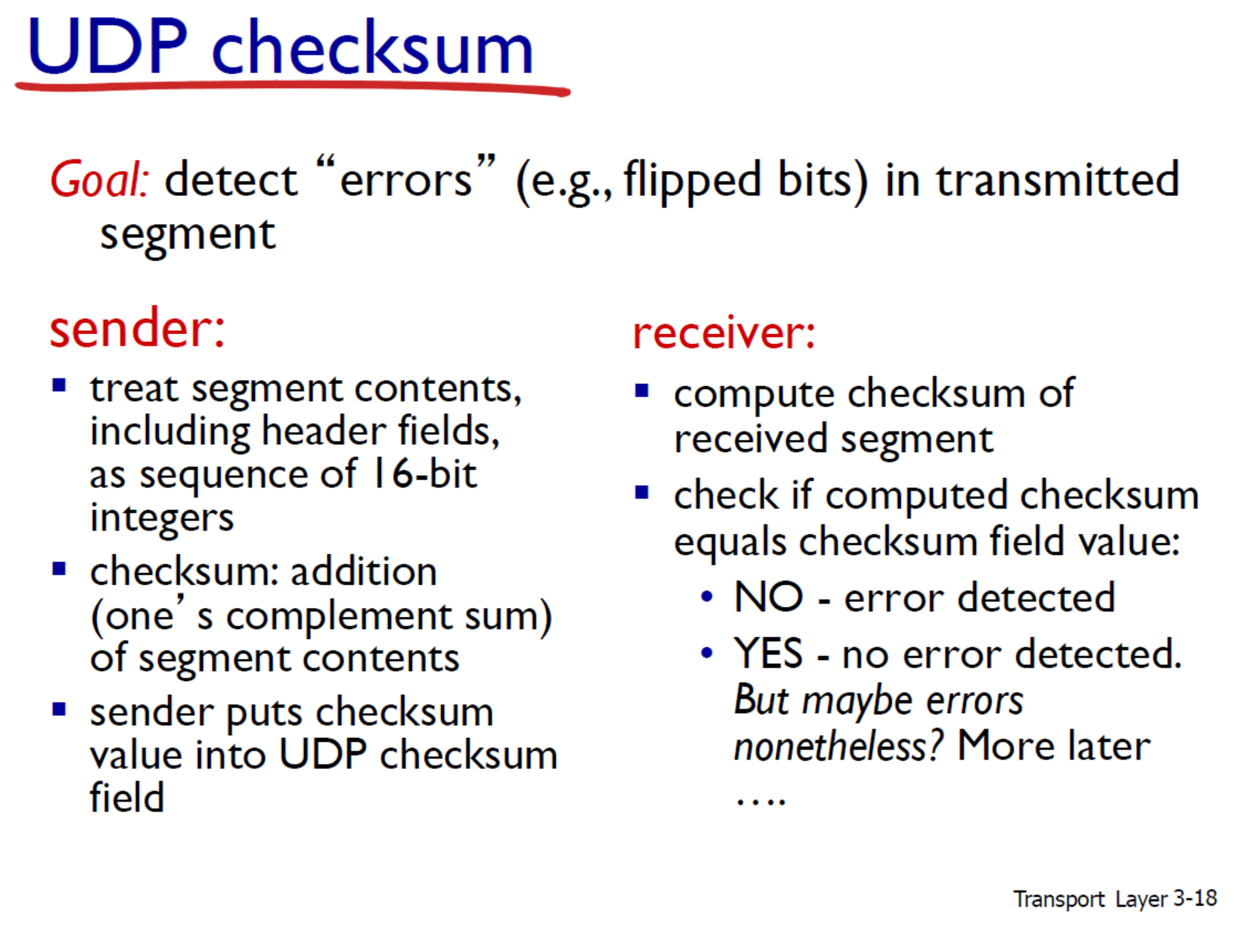
b. Error detection

c. Flow control

d. Congestion control

e. None of the above

UDP uses simple error checking via the checksum field



UDP checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

sender:

- treat segment contents, including header fields, as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?* More later ….

Transport Layer 3-18

A process wants to transmit one byte of data to another process via TCP. Assuming no errors or reordering in the network, how many TCP segments in total would be needed to perform this transmission?

a. 1

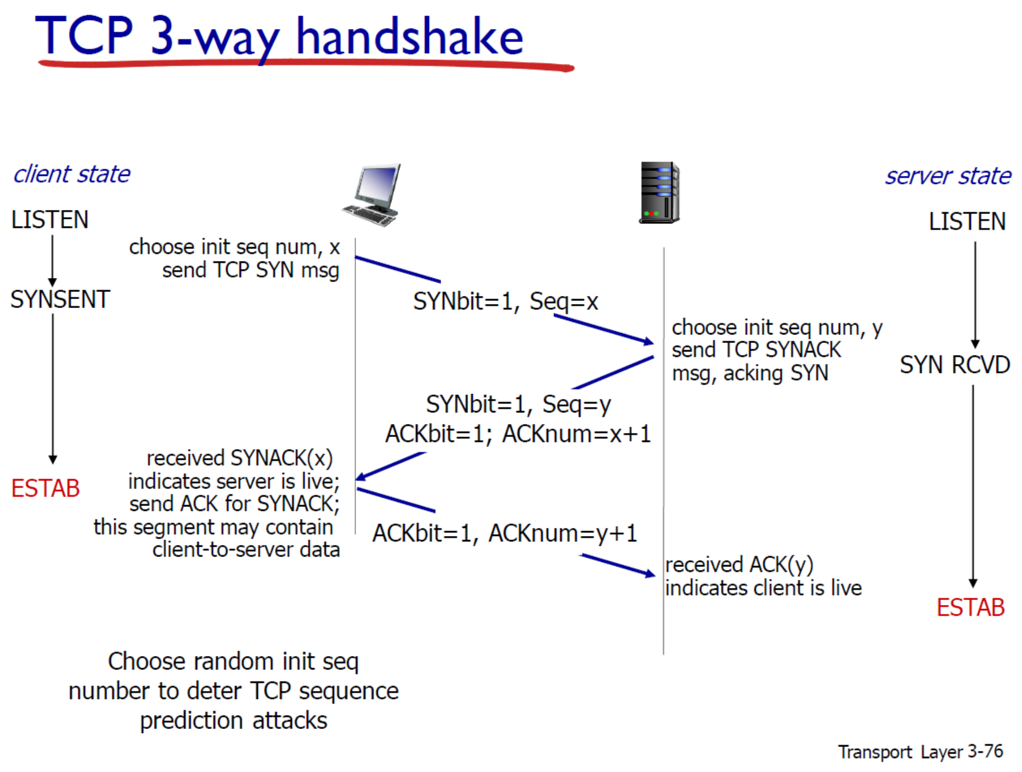
b. 1 or 2

c. 3 or 4

d. 5 or 6

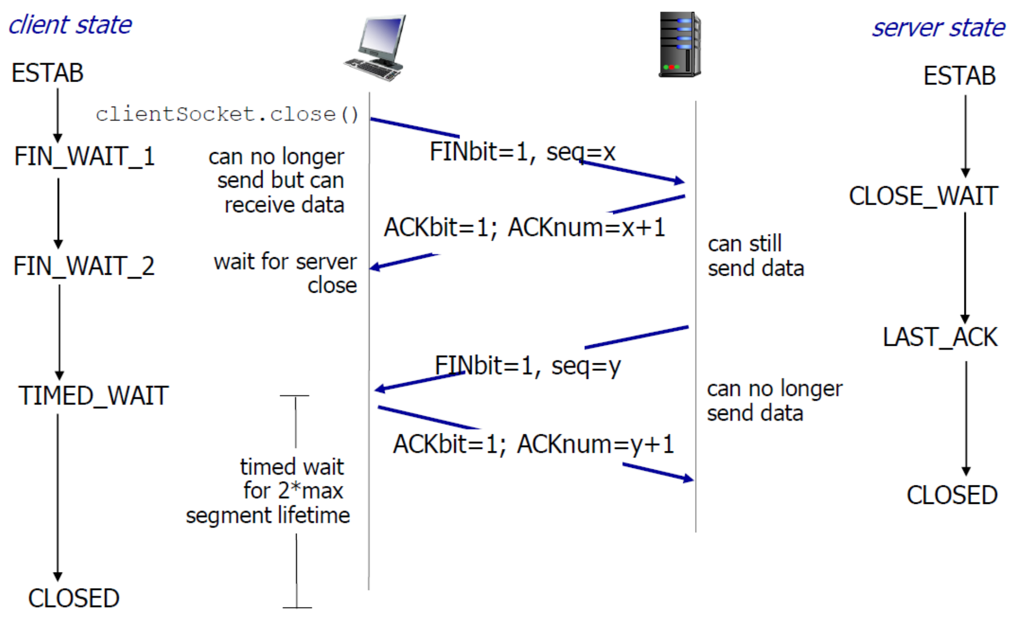
[✓ e. 7 or 8]

Need to consider connection building and tear down.
Three-way TCP handshake, client can include data when ACKing the SYNACK from the server. This process requires 3 segments. The data will require 1 segment if not piggybacked on the SYNACK response. Closing the connection will take 4 segments. In total we have 3 + 1(optional) + 4 = 7 or 8


TCP 3-way handshake
client state
LISTEN
SYNSENT
ESTAB
choose init seq num, x send TCP SYN msg
SYNbit=1, Seq=x
choose init seq num, y send TCP SYNACK msg, acking SYN
SYNbit=1, Seq=y ACKbit=1; ACKnum=x+1
received SYNACK(x) indicates server is live; send ACK for SYNACK; this segment may contain client-to-server data
ACKbit=1, ACKnum=y+1
received ACK(y) indicates client is live
server state
LISTEN
SYN RCVD
ESTAB
Transport Layer 3-76

Choose random init seq number to deter TCP sequence prediction attacks

TCP: closing a connection



Host A is sending two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90 and the second has sequence number 110. Assume both segments have the same data and the segments are received successfully at Host B. How much data is in each segment? What is the acknowledgement number sent to Host A?

a. 90 bytes, ACK = 110

b. 20 bytes, ACK = 110

c. 20 bytes, ACK = 131

d. 20 bytes, ACK = 130

e. None of the above

Remember, we are sending the ACK sequence number for the **next sequence** number expected. In this case, Seq 110 is the start of 20 bytes of data, the last byte is Seq 129 so we will ACK 130.

Consider sending a 2400 byte datagram into a link that has an MTU of 700 bytes. What are the sizes (in bytes) of the 4 fragments including the IP header?

a. 680,680,680,360

b. 700,700,700,340

c. 680,680,680,340

d. 700,700,700,360

e. None of the above

2400 bytes total
2380 bytes (TCP Data) + 20 bytes IP Header – Network Layer
2360 bytes (App Data) + 20 bytes TCP Header – Transport Layer

We need to fragment the 2380 bytes of TCP Data, with an MTU of 700 bytes, this gives us the following:
2380 total to send (we strip off the IP header for fragmentation and work with the TCP segment)

- 700 bytes (fragment #1, 680 bytes of data + 20 bytes IP header)
  2380 minus 680 = 1700 bytes remaining
- 700 bytes (fragment #2, 680 bytes of data + 20 bytes IP header)
  1700 minus 680 = 1020 bytes remaining
- 700 bytes (fragment #3, 680 bytes of data + 20 bytes IP header)
  1020 minus 680 = 340 bytes remaining
- 360 bytes (fragment #4, 340 bytes of data + 20 bytes IP header)
  Complete!