Week 4 Quiz Results for Austin Koske

Score for this attempt: 42 out of 42
Submitted Sep 28 at 2:28pm
This attempt took 18 minutes.

Correct answer

Question 1
2 / 2 pts

3.1-1 Location of transport-layer functionality. Where is transport-layer functionality primarily implemented?

- (selected) Transport layer functions are implemented primarily at the hosts at the “edge” of the network.
- Transport layer functions are implemented primarily at the routers and switches in the network.
- Transport layer functions are implemented primarily at each end of a physical link connecting one host/router/switch to another one host/router/switch.

Nice! Your answer is correct.

Correct answer

Question 2
2 / 2 pts

3.1-2 Transport-layer functionality. True or False: The transport layer provides for host-to-host delivery service?

- True
- (selected) False

Nice! Your answer is correct.

Correct answer

Question 3
2 / 2 pts

3.1-3 Transport layer services using TCP. Check all of the services below that are provided by the TCP protocol.

- [x] Reliable data delivery.
- [x] In-order data delivery
- [ ] A guarantee on the maximum amount of time needed to deliver data from sender to receiver.
- [x] A congestion control service to ensure that multiple senders do not overload network links.
- [ ] A guarantee on the *minimum* amount of throughput that will be provided between sender and receiver.
- [x] A flow-control service that ensures that a sender will not send at such a high rate so as to overflow receiving host buffers.
- [x] A byte stream abstraction, that does not preserve boundaries between message data sent in different socket send calls at the sender.
- [ ] A message abstraction, that preserves boundaries between message data sent in different socket send calls at the sender.

Nice! This answer is correct.

Correct answer

Question 4
2 / 2 pts

3.1-4 Transport-layer services using UDP. Check all of the services below that are provided by the UDP protocol.

- ○ Reliable data delivery.
- ○ In-order data delivery
- ○ A guarantee on the maximum amount of time needed to deliver data from sender to receiver.
- ○ A congestion control service to ensure that multiple senders do not overload network links.
- ○ A guarantee on the *minimum* amount of throughput that will be provided between sender and receiver.
- ○ A flow-control service that ensures that a sender will not send at such a high rate so as to overflow receiving host buffers.
- ○ A byte stream abstraction, that does not preserve boundaries between message data sent in different socket send calls at the sender.
- ◉ A message abstraction, that preserves boundaries between message data sent in different socket send calls at the sender.

Nice! This answer is correct.

Correct answer

Question 5
2 / 2 pts

3.1-5 Network-layer functionality. The transport layer sits on top of the network layer, and provides its services using the services provided to it by the network layer. Thus it's important that we know what is meant by the network layer's "best effort" delivery service. True or False:

> *The network layer's best-effort delivery service means that IP makes its "best effort" to deliver segments between communicating hosts, but it makes no guarantees. In particular, it does not guarantee segment delivery, it does not guarantee orderly delivery of segments, and it does not guarantee the integrity of the data in the segments.*

- (●) Correct! The network layer's best effort service doesn't really provide much service at all, does it?
- ( ) Nope. The network layer's best effort service doesn't really provide much service at all, does it?

Nice! Your answer is correct.

Correct answer

Question 6
2 / 2 pts

3.2-3 Multiplexing/Demultiplexing: UDP port numbers. True or False: When multiple UDP clients send UDP segments to the same destination port number at a receiving host, those segments (from different senders) will always be directed to the same socket at the receiving host.

- (●) True
- ( ) False

Nice. Your answer is correct. UDP demultplexing happens solely on the basis of destination port number. Thus, segments with the same destination port number at a host will have their data demultiplexed to the same socket.

Correct answer

Question 7
2 / 2 pts

3.2-5 Multiplexing UDP with identical port numbers. True or False: It is possible for two UDP segments to be sent from the same socket with source port 5723 at a server to two

different clients.

◉ True

○ False

Nice. Your answer is correct. Recall that the application-layer program explicitly specifies the destination host IP address and port number when sending into a UDP socket, and that a single UDP socket (with a given source port number) can therefore be used to send to multiple UDP clients.

Correct answer

Question 8
2 / 2 pts

3.2-6 Multiplexing TCP with identical port numbers. True or False: It is possible for two TCP segments with source port 80 to be sent by the sending host to different clients.

◉ True

○ False

Nice. Your answer is correct. Recall that two different processes may have different sockets, both associated with the same local port number (say 80, for HTTP) on a host. But, if there are two sockets with the same local port number, how is it that when segments arrive for that host, that data is demultiplexed to the correct port?   Anwser: because TCP demultiplexing happens on the basis of four values (source and destination port numbers, and IP addresses) rather than just on the basis of the destination port number, as is done for UDP.

Correct answer

Question 9
2 / 2 pts

3.3-03 UDP segment length field. Why is the UDP header length field needed?

○ To make the header and even number of bytes

◉ Because the payload section can be of variable length, and this lets UDP know where the segment ends.

○ Because this field is needed in TCP as well.

○ (a) and (b) above

Nice! Your answer is correct.

Correct answer

Question 10
2 / 2 pts

3.3-08 UDP Checksum: how good is it? True or False: When computing the Internet checksum for two numbers, a single flipped bit in each of the two numbers will always result in a changed checksum.

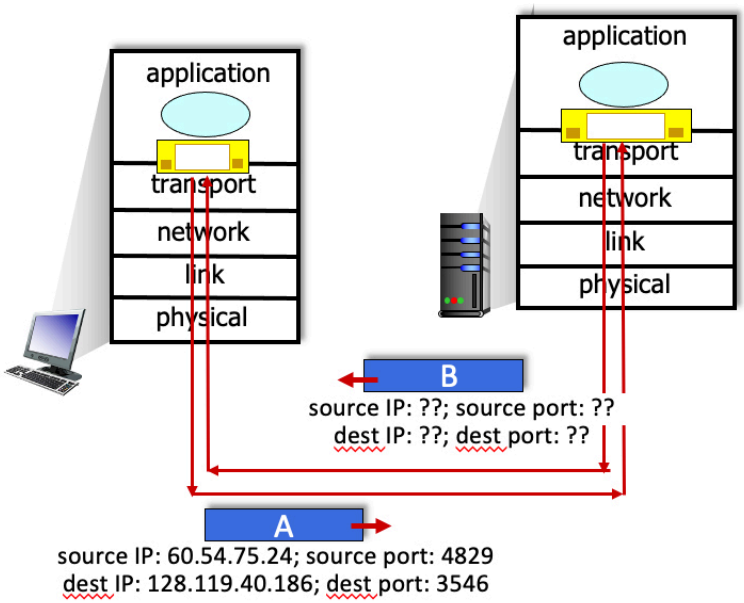
- True
- False

Nice! Your answer is correct.

Correct answer

Question 11
2 / 2 pts

3.3-09 IP addresses and port numbers in a UDP segment sent in reply. Suppose a UDP segment (A in the figure below) arrives at a host with an IP address of 128.119.40.186. The source port in the UDP segment is 4829 and the destination port is 3546. The IP address of the sending host is 60.54.75.24.



Now consider the UDP datagram (and the IP datagram that will encapsulate it) sent in reply by the application on host 128.119.40.186 to the original sender host, labeled B in the figure above. Complete the sentences below ...

What are the source and destination port numbers and IP addresses? (Enter the integer port number or the 4-part dotted decimal IP address, included the period)

The source port number of the UDP segment (B) sent in reply is:

The source IP address of the IP datagram containing the UDP segment (B) sent in reply is:

The destination port number of the UDP segment (B) sent in reply is:

The destination IP address of the IP datagram containing the UDP segment (B) sent in reply is:

[Note: you can find more problems like this one **here** **(http://gaia.cs.umass.edu/kurose_ross/interactive/UDP_Mux_Demux.php)** .]

The source port number of the UDP segment (B) sent in reply is:

3546

The source IP address of the IP datagram containing the UDP segment (B) sent in reply is:

128.119.40.186

The destination port number of the UDP segment (B) sent in reply is:

4829

The destination IP address of the IP datagram containing the UDP segment (B) sent in reply is:

60.54.75.24

Other Incorrect Match Options:

- 80
- 10.0.0.1
- 24

Nice! This answer is correct.

Correct answer

Question 12

2 / 2 pts

3.3-10 How "good" are checksums? Suppose a sender computes a checksum (Internet checksum or some other checksum, which is essentially a sum of the bytes in a segment), puts the checksum in the segment header, and sends the segment to the receiver. The receiver receives the segment (with the checksum in the header). The receiver computes the checksum itself (i.e., performs the same calculation as the sender, but over the received data) and compares the checksum it has computed to the checksum it received in the header. It finds that its computed checksum and received checksum in the header are *identical*. Which of the following statements is true?

- The receiver can be absolutely certain that an error (bit flips) have occurred in the received data in the segment.
- The receiver can be absolutely certain that *no* errors (bit flips) have occurred in the received data in the segment.
- The receiver can't tell for certain whether errors (bit flips) have occurred in the received data in the segment, but can be relatively confident that no errors have occurred.

Nice! Your answer is correct.

Correct answer

Question 13
2 / 2 pts

3.4-06 Cumulative ACK. What is meant by a cumulative acknowledgment, ACK*(n)*?

- A cumulative ACK*(n)* allows the receiver to let the sender know that it has not yet received an ACK for packet with sequence number *n*.
- A cumulative ACK*(n)* allows the receiver to let the sender know that it has not received any packets with a new sequence number since the last cumulative ACK*(n)* was sent.
- A cumulative ACK*(n)* acks all packets with a sequence number up to and including *n* as being received.

Nice. Your answer is correct.

Correct answer

Question 14
2 / 2 pts

3.5-1 TCP reliability semantics. True or False: On the sending side, the TCP sender will take each application-layer chunk of data written into a TCP socket and send it in a distinct TCP segment. And then on the receiving side, TCP will deliver a segment's payload into the appropriate socket, preserving the application-defined message boundary.

- ○ True.
- ◉ False.

Nice. Your answer is right. Remember that TCP's reliability semantics is that of a reliable byte stream - that data bytes will be delivered in-order from sender to receiver. There is no notion of message boundaries in TCP's byte-stream view of the world. This contrasts with UDP, which does preserves application-defined message boundaries.

Correct answer

Question 15
2 / 2 pts

3.5-2 TCP segment format. For the given function of a field in the TCP segment, select the name of that field from the pull-down list.

This field contains the port number associated with the sending socket for this TCP segment.

Source port number

This field contains application data that was written into a socket by the sender of this TCP segment.

Data (or payload).

This field contains the index in the sender-to-receiver byte stream of the first byte of that data in the payload carried in this segment.

Sequence number

This field contains the index in the byte stream of the next in-order byte expected at the receiver

ACK number field

If set, this segment cumulatively ACKs all data bytes up to, but not including, the byte index in the ACK value field of this segment.

ACK bit

This field contains the number of available bytes in the TCP receiver's buffer.

Receiver advertised window

This field contains the Internet checksum of the TCP segment and selected fields in the IP datagram header.

Checksum

This field contains the number of bytes in the TCP header.

Header length field

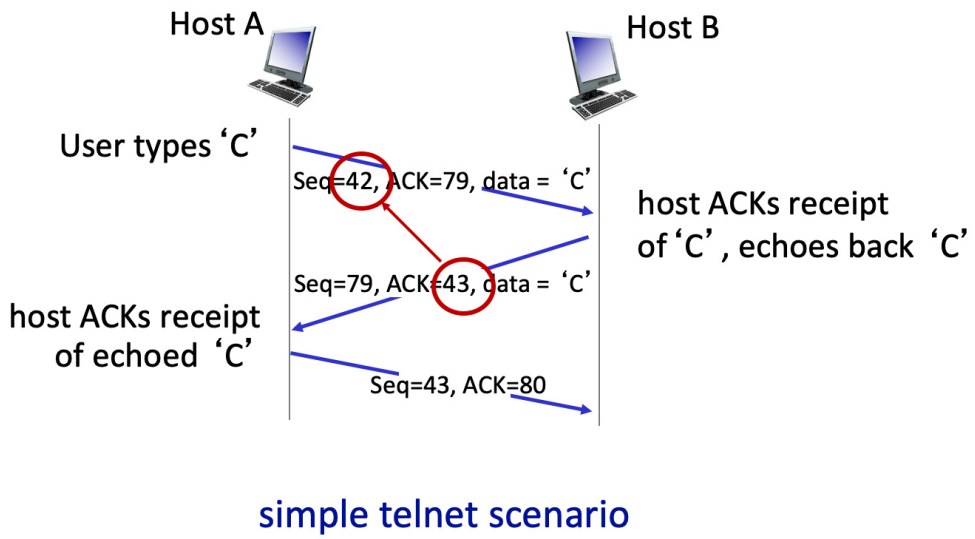
Nice! This answer is correct.

Correct answer

Question 16

2 / 2 pts

3.5-3 TCP sequence numbers and ACKs (1). Consider the TCP Telnet scenario below (from Fig. 3.31 in text). Why is it that the receiver sends an ACK that is one larger than the sequence number in the received datagram?



simple telnet scenario

Because TCP sequence numbers always increase by 1, with every new segment, and the TCP receiver always send the sequence number of the next expected segment

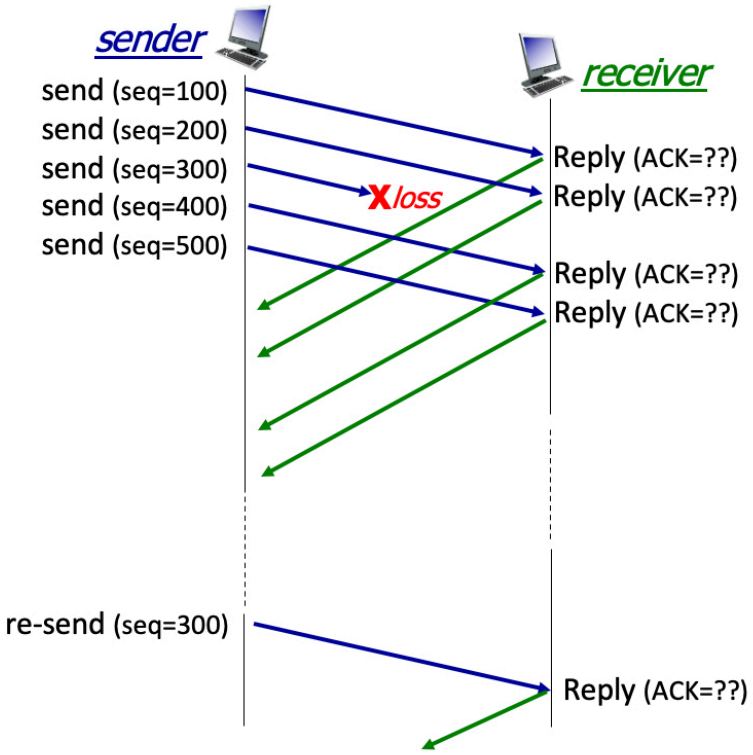
Because the send-to receiver segment carries only one byte of data, and after that segment is received, the next expected byte of data is just the next byte (i.e., has an index that is one larger) in the data stream.

Nice. Your answer is correct. While we numbered segments sequentially in our general development of reliable data transfer, a TCP sequence number is the index of the first byte of data in that TCP segment in the connection's overall byte stream. The ACK number of the index in the stream of the next byte of yet-to-be-received data.

Correct answer

Question 17
2 / 2 pts

3.5-4 TCP sequence numbers and ACKs (2). Suppose that as shown in the figure below, a TCP sender is sending segments with 100 bytes of payload. The TCP sender sends five segments with sequence numbers 100, 200, 300, 400, and 500. Suppose that the segment with sequence number 300 is lost. The TCP receiver will buffer correctly-received but not-yet-in-order segments for later delivery to the application layer (once missing segments are later received).



Complete the sentences below ....

After receiving segment 100, the receiver responds with an ACK with value:

200

After receiving segment 200, the receiver responds with an ACK with value:

300

After receiving segment 500, the receiver responds with an ACK with value:

300, a duplicate ACK

After receiving the *retransmitted* segment 300, the receiver responds with an ACK with value:

600

The TCP receiver does *not* respond in the example, with an ACK with value:

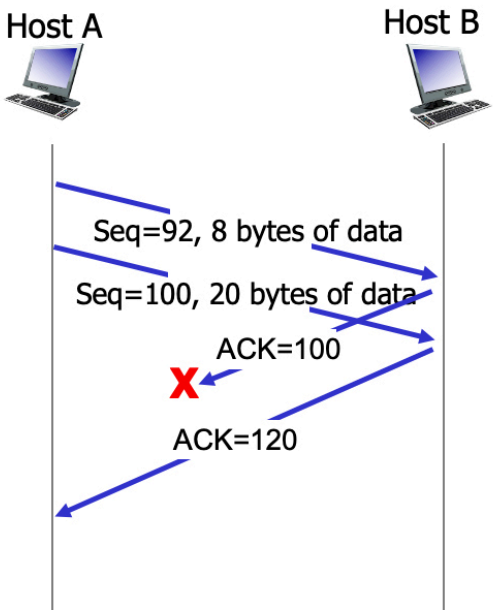
400

Nice! This answer is correct.

Correct answer

Question 18
2 / 2 pts

3.5-6 TCP timer management. Consider the TCP Telnet scenario below (from Fig. 3.36 in text). What timer-related action does the sender take on the receipt of ACK 120?



- Leaves any currently-running timers running.
- Cancels any running timers.
- Restarts a timer for the segment with sequence number 92.

Nice. Your answer is correct. Because of the cumulative nature of the TCP ACK, the receiver knows that the segment with sequence number 92 was received -- even though its ACK was lost. Since there are no unACKed segments, there's no need for any running timers.

Correct answer

Question 19
2 / 2 pts

3.5-7 TCP Flow Control. True or False: with TCP's flow control mechanism, where the receiver tells the sender how much free buffer space it has (and the sender always limits the amount of outstanding, unACKed, in-flight data to less than this amount), it is not possible for the sender to send more data than the receiver has room to buffer.

True

False

Nice! Your answer is correct.

Correct answer

Question 20
2 / 2 pts

3.5-8 TCP connection management. Match the description of a TCP connection management message with the name of the message used to accomplish that function.

A message from client to server initiating a connection request.

SYN message

A message from server to client ACKing receipt of a SYN message and indicating the willingness of the server to establish a TCP connection with the client.

SYNACK message

A message indicating that the sending side is initiating the protocol to terminate a connection.

FIN message

A message sent in response to a request to terminate a connection, ACKing that the side receiving this message is also willing to terminate the connection

FINACK message

A general purpose error message used during connection set up or tear down to let the other side know that an error has occurred, and that the referenced connection should be shut down.

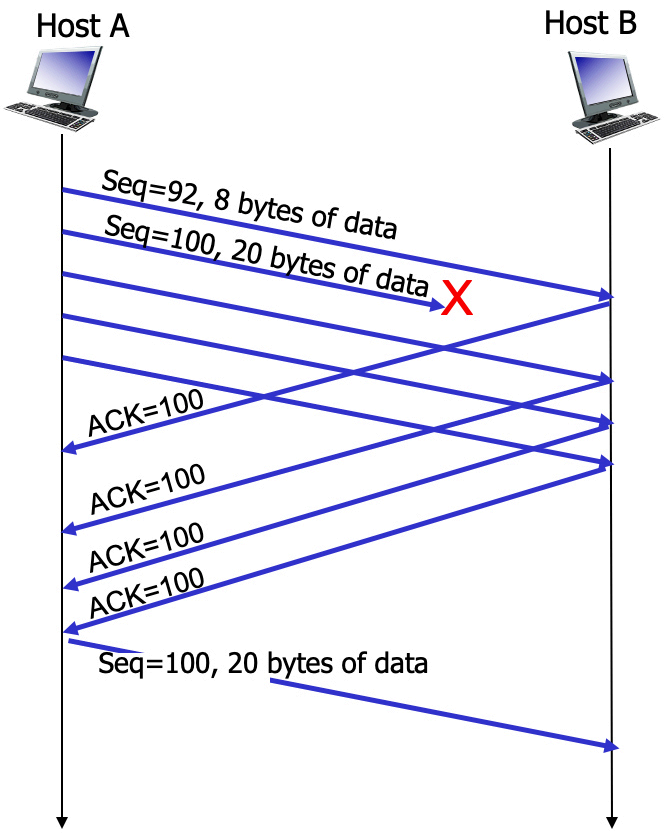
RESET message

Nice! This answer is correct.

Correct answer

Question 21
2 / 2 pts

3.5-9 TCP Fast Retransmit. Consider TCP's Fast Retransmit optimization (see Figure 3.37 from the text, below). Of course, the sender doesn't know for sure that the segment with sequence # 100 is actually lost (it can't see into the channel). Can a sender get three duplicate ACKs for a segment that in fact has *not* been lost? Which of the following statements are true? Suppose a channel can lose, but will not corrupt, messages.



☑ If the channel cannot reorder messages, a triple duplicate ACK indicates to the sender that a segment loss has happened for sure. Actually (again assuming the channel cannot corrupt or reorder messages), even a *single* duplicate ACK would indicate that a segment loss has happed for sure.

☑ If the channel can reorder messages, a triple duplicate ACK can occur even though a message is not lost; since it's possible that a message has just been reordered and has not yet arrived when the three duplicate ACKs were generated.

Nice! This answer is correct.

Quiz Score: 42 out of 42