Introduction and Transport-Layer Services

LOCATION OF TRANSPORT-LAYER FUNCTIONALITY.

Where is transport-layer functionality primarily implemented?

- ○ 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.
- ◉ Transport layer functions are implemented primarily at the hosts at the "edge" of the network.

That's Correct!

CHECK →

1/5

TRANSPORT-LAYER FUNCTIONALITY.

True or False: The transport layer provides for host-to-host delivery service?

- ○ False
- ◉ True.

That's Correct!

← CHECK →

2/5

TRANSPORT LAYER SERVICES USING TCP.

Check all of the services below that are provided by the TCP protocol.

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

That's Correct!

← CHECK →

3/5

TRANSPORT-LAYER SERVICES USING UDP.

Check all of the services below that are provided by the UDP protocol.

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

That's Correct!

← CHECK →

4/5

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?

That's Correct!

← CHECK

5/5

Multi- and Demultiplexing

TRANSPORT-LAYER DEMULTIPLEXING.

What is meant by transport-layer demultiplexing?

- (◉) Receiving a transport-layer segment from the network layer, extracting the payload (data) and delivering the data to the correct socket.
- ( ) Taking data from multiple sockets, all associated with the same destination IP address, adding destination port numbers to each piece of data, and then concatenating these to form a transport-layer segment, and eventually passing this segment to the network layer.
- ( ) Receiving a transport-layer segment from the network layer, extracting the payload, determining the destination IP address for the data, and then passing the segment and the IP address back down to the network layer.
- ( ) Taking data from one socket (one of possibly many sockets), encapsulating a data chuck with header information – thereby creating a transport layer segment – and eventually passing this segment to the network layer.

That's Correct!

CHECK →

1/6

TRANSPORT-LAYER MULTIPLEXING.

What is meant by transport-layer multiplexing?

- ( ) Receiving a transport-layer segment from the network layer, extracting the payload (data) and delivering the data to the correct socket.
- ( ) Taking data from multiple sockets, all associated with the same destination IP address, adding destination port numbers to each piece of data, and then concatenating these to form a transport-layer segment, and eventually passing this segment to the network layer.
- ( ) Receiving a transport-layer segment from the network layer, extracting the payload, determining the destination IP address for the data, and then passing the segment and the IP address back down to the network layer.
- (◉) Taking data from one socket (one of possibly many sockets), encapsulating a data chuck with header information – thereby creating a transport layer segment – and eventually passing this segment to the network layer.

That's Correct!

← CHECK →

2/6

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.

- ( ) False
- (◉) True

That's Correct!

← CHECK →

3/6

MULTIPLEXING/DEMULTIPLEXING: TCP PORT NUMBERS.

True or False: When multiple TCP clients send TCP 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.

- (◉) False
- ( ) True

That's Correct!

← CHECK →

4/6

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

That's Correct!

← CHECK →

5/6

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

That's Correct!

← CHECK

6/6

Connectionless Transport: UDP

DOES UDP PRESERVE APPLICATION-LAYER MESSAGE BOUNDARIES?

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

- (x) True
- ( ) False

That's Correct!

CHECK →

1/10

UDP HEADER FIELDS.

Which of the fields below are in a UDP segment header? *[Hint: note the use of the word "header" in this question statement.]*

- [x] Source port number
- [x] Destination port number
- [ ] Sequence number
- [ ] Source IP address
- [ ] Data (payload)
- [ ] Upper layer protocol
- [x] Length (of UDP header plus payload)
- [x] Internet checksum

Note: IP addresses are in the IP segment

That's Correct!

← CHECK →

2/10

UDP SEGMENT LENGTH FIELD.

Why is the UDP header length field needed?

- ( ) To make the header and even number of bytes
- (x) Because the payload section can be of variable length, and this lets UDP know where the segment ends.
- ( ) (a) and (b) above
- ( ) Because this field is needed in TCP as well.

That's Correct!

← CHECK →

3/10

INTERNET CHECKSUM AND UDP.

Over what set of bytes is the checksum field in the UDP header computed over?

- ( ) The entire UDP segment, except the checksum field itself.
- (x) The entire UDP segment, except the checksum field itself, and the IP sender and receive address fields
- ( ) Just the UDP header but not the payload.

That's Correct!

← CHECK →

4/10

WHAT IS A CHECKSUM?

Which of the following statements are true about a checksum? Hint: more than one statement is true.

- [x] The receiver of a packet with a checksum field will add up the received bytes, just as the sender did, and compare this locally-computed checksum with the checksum value in the packet header. If these two values are *different* then the receiver*knows* that one of the bits in the received packet has been changed during transmission from sender to receiver.
- [x] A checksum is computed at a sender by considering each byte within a packet as a number, and then adding these numbers (each number representing a bytes) together to compute a sum (which is known as a checksum).
- [x] The sender-computed checksum value is often included in a checksum field within a packet header.
- [ ] The receiver of a packet with a checksum will add up the received bytes, just as the sender did, and compare this locally-computed checksum with the checksum value in the packet header. If these two values are the *same* then the receiver *knows*that all of the bits in the received packet are correct, i.e., that no bits have been changed during transmission from sender to receiver.

That's Correct!

← CHECK →

5/10

COMPUTING THE INTERNET CHECKSUM (1).

Compute the Internet checksum value for these two 16-bit words: 11110101 11010011 and 10110011 01000100

[Note: you can find more problems like this one here.]

- ○ 01011110 11000101
- ○ 01101110 11010101
- ◉ 01010110 11100111
- ○ 01010110 11101000

That's Correct!

← CHECK →

Num1		1	1	1	1	0	1	0	1	1	1	0	1	0	0	1	1
Num2		1	0	1	1	0	0	1	1	0	1	0	0	0	1	0	0
Sum	1	1	0	1	0	1	0	0	1	0	0	0	1	0	1	1	1
Carry		1	0	1	0	1	0	0	1	0	0	0	1	1	0	0	0
Flip		0	1	0	1	0	1	1	0	1	1	1	0	0	1	1	1

COMPUTING THE INTERNET CHECKSUM (2).

Compute the Internet checksum value for these two 16-bit words: 01000001 11000100 and 00100000 00101011

[Note: you can find more problems like this one here.]

- ○ 01101110 11010101
- ◉ 10011110 00010000
- ○ 10011110 00001111
- ○ 10011110 00010001

That's Correct!

← CHECK →

Num1		0	1	0	0	0	0	0	1	1	1	0	0	0	1	0	0
Num2		0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	1
Sum		0	1	1	0	0	0	0	1	1	1	1	0	1	1	1	1
Carry																	
Flip		1	0	0	1	1	1	1	0	0	0	0	1	0	0	0	0

UDP CHECKSUM: HOW GOOD IS IT?

True or False: When computing the Internet checksum for two numbers, a single flipped bit (i.e., in just one of the two numbers) will always result in a changed checksum.

- ◉ True
- ○ False

That's Correct!

← CHECK →

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.

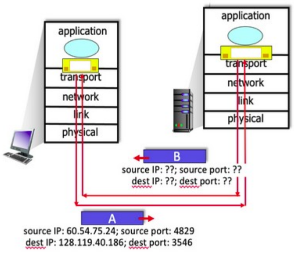
- ○ True
- ◉ False

That's Correct!

← CHECK →

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.]

QUESTION LIST:

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

E

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

D

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

F

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

A

ANSWER LIST:

A. 60.54.75.24

B. 80

C. 10.0.0.1

D. 128.119.40.186

E. 3546

F. 4829

G. 24

That's Correct!

← CHECK

10/10

Principles of Reliable Data Transfer

QUESTION LIST:

Lets the sender know that a packet was NOT received correctly at the receiver.

E

Used by sender or receiver to detect bits flipped during a packet's transmission.

B

Allows for duplicate detection at receiver.

C

Lets the sender know that a packet was received correctly at the receiver.

D

Allows the receiver to eventually receive a packet that was corrupted or lost in an earlier transmission.

A

ANSWER LIST:

A. Retransmission

B. Checksum

C. Sequence numbers

D. ACK

E. NAK

That's Correct!

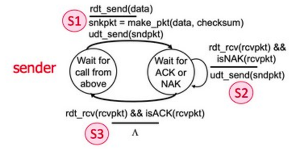
CHECK →

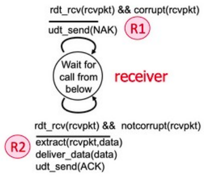
1/14

THE RDT 2.0 PROTOCOL.

Consider the rdt 2.0 sender and receiver shown below, with FSM transitions at the sender labeled S1, S2, and S3; and receiver transitions labeled R1 and R2.

Which of the following sequences of transitions could possibly occur as a result of an initial rdt_send() call at the sender, and possible later message corruption and subsequent error recovery.





Receiver sends ACK (R2)
Sender can't have received a NAK (S2)

- [x] S1, R1, S2, R2,S3
- [x] S1, R2, S3
- [ ] S1, R2, S2
- [ ] S1, R1, S3
- [ ] S1, S2, S3
- [x] S1, R1, S2
- [ ] S1, R1, S2, R1, S3

Can't skip S2

Receiver not involved

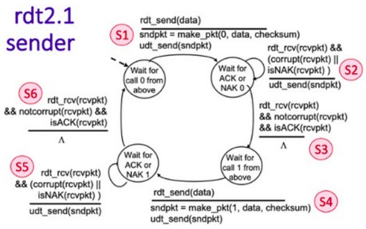
That's Correct!

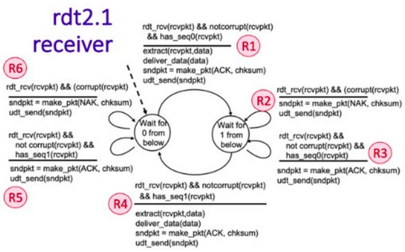
← CHECK →

Receiver sends NAK(second R1)
Sender can't have received an ACK (S3)

THE RDT 2.1 PROTOCOL (A).

Consider the rdt2.1 sender and receiver FSMs shown below, with labeled transitions S1 through S6 at the sender, and transitions R1 through R6 at the receiver. The sender and receiver start in the "Wait for call 0 from above" and "Wait for 0 from below" states, respectively.





Suppose that no channel errors occur. A sequence of interleaved sender and receiver transitions is given below. Transitions S1 and S4 are already provided. Choose the sender or receiver transition for the unlabeled transitions x_1, x_2, x_3, and x_4 below to indicate the time-ordered sequence of transitions (interleaved sender and receiver transitions) that will result in two messages being delivered at the receiver, with the sender and receiver returning to their initial states (again, given that no channel errors occur).

S1, x_1, x_2, S4, x_3, x_4

QUESTION LIST:

transition x_1

transition x_2

transition x_3

transition x_4

ANSWER LIST:

A. S2

B. R4

C. R3

D. S3

E. S6

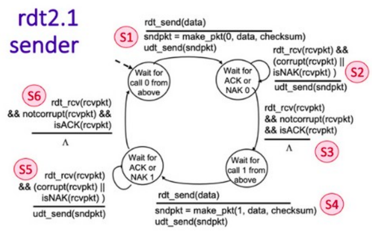
F. R1

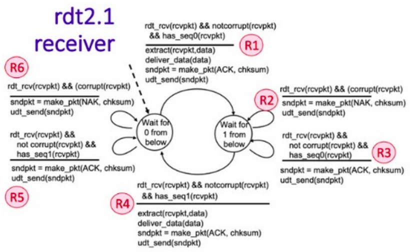
That's Correct!

← CHECK →

THE RDT 2.1 PROTOCOL (B).

Consider the rdt2.1 sender and receiver FSMs shown below, with labeled transitions S1 through S6 at the sender, and transitions R1 through R6 at the receiver. The sender and receiver start in the "Wait for call 0 from above" and "Wait for 0 from below" states, respectively.





Suppose that the initial message transmission by the sender is corrupted, but that no other message transmissions are corrupted. Match the unlabeled transitions x_1, x_2, x_3, x_4, x_5 in the time-ordered sequence of transitions below (interleaved sender and receiver transitions) that will occur following the initial S1 transition (which is corrupted), that will result in two messages being delivered at the receiver, with the sender and receiver returning to their initial states (again, given that the initial message transmission by the sender is corrupted). Note that transitions S1, S4, and S6 are already provided below.

S1 (message corrupted), x_1, x_2, x_3, x_4, S4, x_5, S6.

QUESTION LIST:

transition x_1

transition x_2

transition x_3

transition x_4

transition x_5

ANSWER LIST:

A. S3

B. R6

C. S2

D. R4

E. S4

F. R3

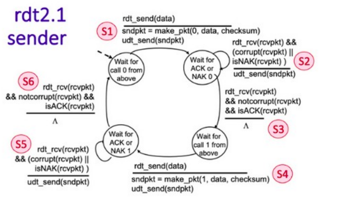
G. R1

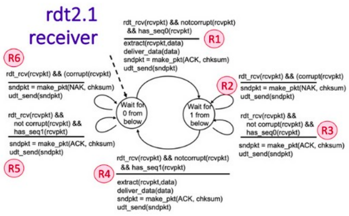
That's Correct!

← CHECK →

THE RDT 2.1 PROTOCOL (C).

Consider the rdt2.1 sender and receiver FSMs shown below, with labeled transitions S1 through S6 at the sender, and transitions R1 through R6 at the receiver. The sender and receiver start in the "Wait for call 0 from above" and "Wait for 0 from below" states, respectively.





Suppose that the first packet from the sender is correctly received at the receiver but that ACK message sent from receiver-to-sender is corrupted; all other messages (before or after that ACK) are transmitted error-free. Match the unlabeled transitions x_1, x_2, x_3, x_4, x_5 in the time-ordered sequence of transitions below (interleaved sender and receiver transitions) that will occur following the initial S1 transition, which is followed by a corrupted ACK transmission, that will result in a message being delivered at the receiver, with the sender and receiver returning to their initial states. Note that some transitions are already provided below.

S1, x_1 $_{(ACK\ corrupted)}$, x_2, x_3, x_4, S4, x_5, S6.

QUESTION LIST:

transition x_1

transition x_2

transition x_3

transition x_4

transition x_5

ANSWER LIST:

A. R1

B. R4

C. R3

D. R2

E. S2

F. S3

That's Correct!

← CHECK →

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)* acks all packets with a sequence number up to and including *n* as being received.
- ○ 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.

That's Correct!

← CHECK →

6/14

STOP-AND-WAIT: CHANNEL UTILIZATION.

Suppose a packet is 10K bits long, the channel transmission rate connecting a sender and receiver is 10 Mbps, and the round-trip propagation delay is 10 ms. What is the maximum channel utilization of a stop-and-wait protocol for this channel?

- ◉ .1
- ○ .001
- ○ .01
- ○ 10.0
- ○ 1.0

That's Correct!

← CHECK →

7/14

Calculation: $$U=\frac{L/R}{RTT+L/r}=\frac{\dfrac{10\,kbits}{10\,Mbps*1000}}{\dfrac{10\,ms}{1000}+\dfrac{10\,kbits}{10\,Mbps*1000}}$$

CHANNEL UTILIZATION WITH PIPELINING.

Suppose a packet is 10K bits long, the channel transmission rate connecting a sender and receiver is 10 Mbps, and the round-trip propagation delay is 10 ms. What is the channel utilization of a pipelined protocol with an arbitrarily high level of pipelining for this channel?

- ○ 10.0
- ◉ 1.0
- ○ 0.1
- ○ 0.01
- ○ 0.001

High level of pipelining: We can assume we are making FULL (100%=1.0) utilization

That's Correct!

← CHECK →

CHANNEL UTILIZATION WITH PIPELINING (MORE).

Suppose a packet is 10K bits long, the channel transmission rate connecting a sender and receiver is 10 Mbps, and the round-trip propagation delay is 10 ms. How many packets can the sender transmit before it starts receiving acknowledgments back?

- ○ 10,000
- ○ 100
- ○ 1
- ○ 1000
- ◉ 10

That's Correct!

← CHECK →

9/14

How long does it take to receive an ack?

$$\boxed{\left(\frac{10\,kbits}{10\,Mbps*1000}\right)+\left(\frac{10\,ms}{1000}\right)} = 0.011 seconds$$

In 0.011seconds, how many packets could be transferred (assume pipelined)?

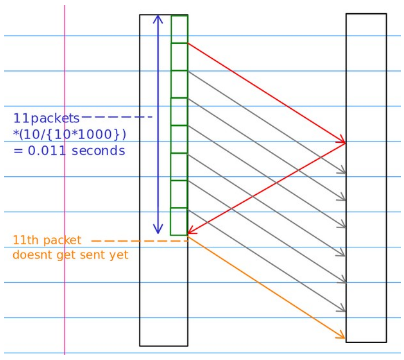
Each packet takes $\boxed{\frac{10\,kbits}{10\,Mbps*1000}}$ *= 0.001 seconds to be sent into the link*

So if x is number of packets, then $\boxed{x*0.001}$ $=0.011$

Therefore 11 packets sent in 0.011 seconds

But 11th packet would have been transmitted at exactly t=0.011s

Therefore only the first 10 packets would have actually been ***sent*** *(note: sent does not necessarily mean received as well)*



PIPELINING.

Which of the following statements about pipelining are true? One or more statements may be true.

- [x] A pipelined sender can have transmitted multiple packets for which the sender has yet to receive an ACK from the receiver.
- [x] With a pipelined sender, there may be transmitted packets "in flight" – propagating through the channel – packets that the sender has sent but that the receiver has not yet received.
- [ ] With pipelining, a receiver will have to send fewer acknowledgments as the degree of pipelining increases
- [ ] With pipelining, a packet is only retransmitted if that packet, or its ACK, has been lost.

Only the case with Go-back-N

What if packet is corrupt?

That's Correct!

← CHECK →

10/14

PACKET BUFFERING IN GO-BACK-N.

What are some reasons for discarding received-but- out-of-sequence packets at the receiver in GBN? Indicate one or more of the following statements that are correct.

- [ ] Discarding an out of sequence packet will really force the sender to retransmit.
- [ ] If some packets are in error, then its likely that other packets are in error as well.
- [x] The sender will resend that packet in any case.
- [x] The implementation at the receiver is simpler.

That's Correct!

← CHECK →

11/14

PACKET BUFFERING IN GO-BACK-N (MORE).

What are some reasons for *not* discarding received-but- out-of-sequence packets at the receiver in GBN? Indicate one or more of the following statements that are correct.

- [ ] Complex protocols are always better.
- [x] Even though that packet will be retransmitted, its next retransmission could be corrupted, so don't discard a perfectly well-received packet, silly!
- [ ] By not discarding, the receiver can implicitly let the sender know that it (the sender) does not necessarily have to retransmit that packet.

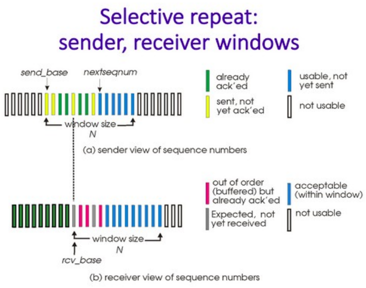
That's Correct!

← CHECK →

12/14

RECEIVER OPERATION IN SELECTIVE REPEAT.

In the SR receiver window (see diagram below, taken from PPT slides and video), why haven't the red packets been delivered yet? Check the one or more reasons below that apply.



"delivery" = The transport layer pushing up to the application layer

- [ ] There is a packet with a higher sequence number than any of the red packets that has yet to be received, so in-order delivery of data in the red packets to the application layer is not yet possible.
- [ ] Red packets have a lower delivery priority up to the application.
- [x] There is a packet with a lower sequence number than any of the red packets that has yet to be received, so in-order delivery of data in the red packets up to the application layer is not yet possible.

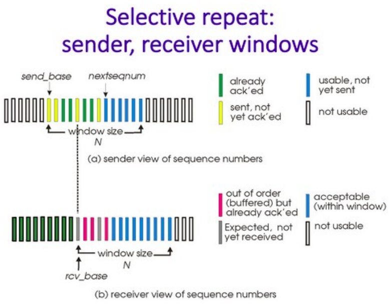
That's Correct!

← CHECK →

13/14

RECEIVER OPERATION IN SELECTIVE REPEAT (MORE).

In SR, why does the receiver have to acknowledge packets with sequence numbers that are less than (and to the left of) those in its window, which starts at *rcv_base*.



- [ ] Because, at the time of the data packet arrival at the receiver, the sender has definitely still not received an ACK for that packet.
- [ ] Actually, this ACK retransmission can be ignored and the protocol will still function correctly, but its performance won't be as good.
- [x] Because the sender may not have received an ACK for that packet yet.

That's Correct!

← CHECK

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.

That's Correct!

CHECK →

TCP does not care about the separation between messages. It just sends clumps together, which may even be pieces of different messages

UDP preserves message boundaries, not TCP

QUESTION LIST:

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

F

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

H

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.

D

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

B

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.

E

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

G

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

C

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

A

ANSWER LIST:

A. Header length field

B. ACK number field

C. Checksum

D. Sequence number

E. ACK bit

F. Source port number

G. Receiver advertised window

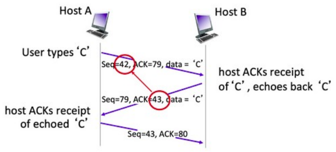
H. Data (or payload).

That's Correct!

← CHECK →

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

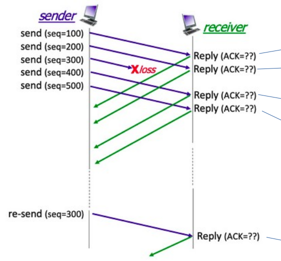
- ○ Because TCP sequence numbers always increase by 1, with every new segment, and the TCP receiver always send the sequence number of the next expected segment
- ◉ Because the send-to receiver segment carries only one byte of data, and after that segment is received, the next expected byte of data is just the next byte (i.e., has an index that is one larger) in the data stream.

That's Correct!

← CHECK →

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).



Received 100, ACK 200

Received 200, ACK 300

Receives 400, stores it, but then
"Excuse me, I didn't receive 300" --> ACK 300

Receives 500, stores it, but then
"Excuse me, I didn't receive 300" --> ACK 300

ACK 600

Complete the sentences below ....

QUESTION LIST:

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

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

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

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

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

ANSWER LIST:

A. 200

B. 600

C. 300, a duplicate ACK

D. 300

E. 400

That's Correct!

← CHECK →

TCP RTT ESTIMATION: EWMA.

Consider TCP use of an exponentially weighted moving average (EWMA) to compute the nth value of the estimated RTT:

$EstimatedRTT_n$ = (1- a)$EstimatedRTT_{n-1}$ + a*$SampleRTT_n$*

True or False: with this EWMA algorithm the value of *$EstimatedRTT_n$* has no dependence on the earlier sample, *$SampleRTT_{n-1}$*

○ True

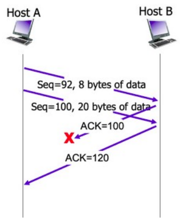
◉ False

That's Correct!

← CHECK →

5/9

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?



Ack 120 means no problem with number 92, so dont care anymore (dont need to restart the timer, and dont need to leave it running)

○ Leaves any currently-running timers running.

◉ Cancels any running timers.

○ Restarts a timer for the segment with sequence number 92.

That's Correct!

← CHECK →

6/9

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.

○ False

◉ True

That's Correct!

← CHECK →

7/9

QUESTION LIST:

A message from client to server initiating a connection request.

E

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.

A

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

C

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

B

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.

D

ANSWER LIST:

A. SYNACK message

B. FINACK message

C. FIN message

D. RESET message

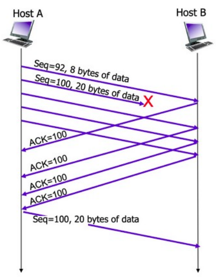
E. SYN mesage

That's Correct!

← CHECK →

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.

That's Correct!

← CHECK

Congestion Control

CONGESTION CONTROL VERSUS FLOW CONTROL.

Consider the five images below. Indicate which of these images suggest the need for *flow* control (the others would suggest the need for congestion control).

- [ ] A crowd of people
- [ ] Car traffic
- [ ] A penguin crowd
- [x] A talking head
- [x] A glass overflowing

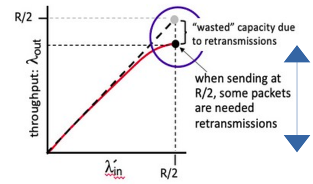
That's Correct!

CHECK →

1/5

TWO CONGESTED SENDERS.

Consider the figure below, which shows the application-to-application throughput achieved when two senders are competing at a shared bottleneck link. Suppose that when the overall arrival rate, $lambda_{in}'$ (for each sender) is close to R/2, the throughput to the application layer (at each receiver), $lambda_{out}$, is equal to 0.8 * $lambda_{in}'$.



This distance is 0.8*lambda(in)
So to get wasted capacity:
1*lambda(in)-0.8*lambda(in)

What fraction of the packets transmitted at the sender are retransmissions?

- (•) .20
- ( ) .80
- ( ) 0
- ( ) .50

That's Correct!

← CHECK →

2/5

NETWORK-ASSISTED OR END-END CONGESTION CONTROL?

Which of the following actions are used in network-assisted congestion control (say versus end-end congestion control) to signal congestion. Check all that apply.

- [x] A router sends an ICMP message to a host telling it to slow down its sending rate.
- [ ] A datagram experiences delay at a congested network router, which is then measured by the sender and used to decrease the sending rate.
- [ ] The transport-layer receiver informs sender of the size of its (transport-payer receiver) receive window.
- [ ] The sender decreases its sending rate in response to a measured increase in the RTT.
- [ ] A sender decreases its sending rate in response to packet loss detected via its transport-layer ACKing.
- [ ] A router drops a packet at a congested router.
- [x] A router marks a field in the datagram header at a congested router.

That's Correct!

← CHECK →

3/5

NETWORK-ASSISTED OR END-END CONGESTION CONTROL (2)?

Which of the following actions are associated with end-end congestion control (say versus network-assisted congestion control). Check all that apply.

- [x] A datagram experiences delay at a congested network router, which is then measured by the sender and used to decrease the sending rate.
- [x] A sender decreases its sending rate in response to packet loss detected via its transport-layer ACKing.
- [x] The transport-layer sender decreases its sending rate in response to a measured increase in the RTT.
- [ ] A router marks a field in the datagram header at a congested router.
- [x] A router drops a packet at a congested router, which causes the transport-layer sender to infer that there is congestion due to the missing ACK for the lost packet.
- [ ] A router sends an ICMP message to a host telling it to slow down its sending rate.
- [ ] The transport-layer receiver informs sender of the size of its (transport-payer receiver) receive window.

That's Correct!

← CHECK →

4/5

DIFFERENT APPROACHES TOWARDS CONGESTION CONTROL.

Use the pulldown menu to match a congestion control approach to how the sender detects congestion.

QUESTION LIST:

The sender infers segment loss from the absence of an ACK from the receiver.

B

Bits are set at a congested router in a sender-to-receiver datagram, and bits are in the returned to the sender in a receiver-to sender ACK, to indicate congestion to the sender.

C

The sender measures RTTs and uses the current RTT measurement to infer the level of congestion.

A

ANSWER LIST:

A. delay-based

B. end-end

C. network-assisted

That's Correct!

← CHECK

5/5

TCP'S AIMD ALGORITHM.

Which of the following statements about TCP's Additive-increase-multiplicative-decrease (AIMD) algorithm are true? Check all that are true.

- [x] AIMD cuts the congestion window size, cwnd, in half whenever loss is detected by a triple duplicate ACK.
- [ ] AIMD *always* cuts the congestion window size, cwnd, in half whenever loss is detected.
- [x] AIMD is a end-end approach to congestion control.
- [x] AIMD cuts the congestion window size,cwnd, i to 1 whenever a timeout occurs.
- [ ] AIMD uses the measured RTT delay to detect congestion.
- [ ] AIMD is a network-assisted approach to congestion control.
- [ ] AIMD uses observed packet loss to detect congestion.

That's Correct!

CHECK →

1/6

TCP'S AIMD ALGORITHM (2).

How is the sending rate typically regulated in a TCP implementation?

- ( ) By using the retransmission timeout timer and counting the number of bytes sent since the last timeout to compute the sending rate since that last timeout, and then making sure its sending rate never exceed the rate set by AIMD.
- (•) By keeping a window of size cwnd over the sequence number space, and making sure that no more than cwnd bytes of data are outstanding (i.e, unACKnowledged). The size of cwnd is regulated by AIMD.

That's Correct!

← CHECK →

2/6

TCP'S SLOWSTART ALGORITHM.

Which of the following best completes this sentence: "In the absence of loss, TCP slow start increases the sending rate ... "

- ( ) ... slower than AIMD, that's why it's called Slowstart."
- (•) " ... faster than AIMD. In fact, slowstart increases the sending rate exponentially fast per RTT."
- ( ) "... at the same rate as AIMD."

That's Correct!

← CHECK →

3/6

UNCONTROLLED TRANSPORT-LAYER SENDERS.

Consider the transport-layer flows interacting at a congested link. In the face of such congestion, what happens at this link to a transport-layer flow that does not cut back on its sending rate?

- (•) Nothing different from the other flows crossing the congested link.
- ( ) That sender's datagrams will be preferentially dropped at the congested link.
- ( ) The router will send a signal to the TCP sender that would force the TCP sender to cut its rate in half.

That's Correct!

← CHECK →

4/6

TCP CUBIC.

Assuming that the congestion window size,cwnd, has not yet reached W_{max}, TCP CUBIC will ... (check all that apply)

- [ ] ... have a sending rate that always increases faster than that of AIMD.
- [x] ... always have a window size, cwnd, and hence a sending rate, higher than that of AIMD (assuming a given window size, W_{max}, at which loss would occur).
- [x] ... increase its sending rate faster than AIMD when cwnd is far away from W_{max}, but increase slower than AIMD when cwnd is closer to W_{max}

That's Correct!

← CHECK →

5/6

QUESTION LIST:

The currently measured throughput is greater than cwnd/RTT_{min}

B

The currently measured throughput is equal to or a bit less than than cwnd/RTT_{min}

C

The currently measured throughput is much less that than cwnd/RTT_{min}

A

ANSWER LIST:

A. decrease the sending rate

B. This should never happen.

C. increase the sending rate

That's Correct!

← CHECK