Questions: 6,15,17,22,25,27,37,40,44

6. 1. S state: Wait for call 0 from above, R state: Wait for 0 from below
   S sends pkt with sequence number 0
   2. S state: Wait for ACK or NAK 0, R state: Wait for 0 from below
   R receives pkt with sequence number 0
   3. S state: Wait for ACK or NAK 0, R state: Wait for 0 from below
   R sends ACK with sequence number 0 (corrupted)
   4. S state: Wait for ACK or NAK 0, R state: Wait for 1 from below
   S receives ACK with sequence number 0 (corrupted)
   5. S state: Wait for ACK or NAK 0, R state: Wait for 1 from below
   S resends pkt with seq number 0, when R expects pkt 1
   6. S keeps sending pkt 0 and the receiver keeps sending NAK 1
   Deadlock state

15. $L = 1500\,B = 1500 * 8\,b = 12000\,b$

$R = 100\,Mbps = 100 * 10^{6}\,bps$

$d_{prop} = 25\,ms$

$RTT = 2 * d_{prop} = 2 * 25ms = 50ms$

$d_{trans} = L/R = 12000\,b/10^{8}bps = 0.00012\,s = .12\,ms$

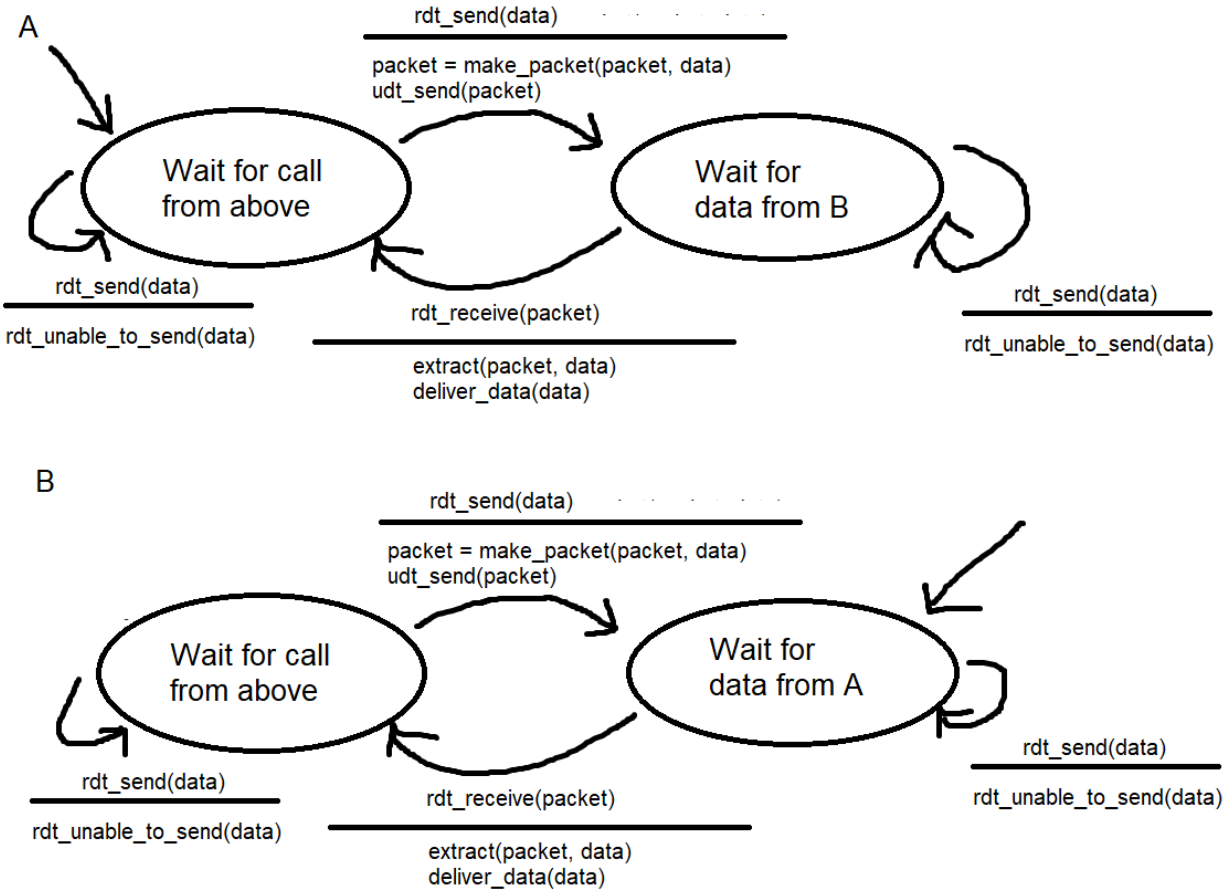
$U = W * \frac{L/R}{L/R + RTT} \rightarrow .98 = W * \frac{.12ms}{.12ms + 50ms} \rightarrow .98 = W * 0.00239$

$.98/0.00239 = W$

$W = 409.313$

To reach 98% utilization, a window size of 410 packets is needed for this link (409 would be a lower utilization than 98%)

17.



22. A.

If the current packet k is the first in the sender's window (The sender has received ACK for all k-1 packets), then the range of sequence numbers in the sender's window is [k, k-1+ 4] since 4 is the window size.
If the current packet k is last in the sender's window, then the range of sequence numbers in the sender's window is [k-4+1, k]
Thus, the possible range of sequence numbers in the sender's window at time t is [k-4+1, k-1+4] meaning [k-3, k+3]

B.

If packet k is the first in the window, then the previous W packets were ACKed, but these ACKs may not have reached the sender at time t, so the range of inflight ACK sequence numbers starts at k-4, and if packet k is not the first in a window, then k-1 is the most recent ACK that may not have been received yet, so the total range of ACK sequence numbers in the pipeline is [k-4, k-1]

25. A.

UDP has more control of what data is sent in a packet because UDP encapsulates the message given to it by the application, so a message does not

travel in the same segment as another message. TCP does not follow this pattern, and multiple messages may blend together depending on how buffering/queueing processes decide.

B.

UDP has more control over when the data is sent because UDP does not use buffering. It gets the data a sends it, not needing any overhead to establish connections or wait for ACK/NAK. UDP does not experience congestion or need to adjust for flow control, which can slow down TCP segments substantially.

27.

A.

If host B has already received bytes up to 126, then next byte expected is byte 127. The sequence number of the first segment is 127 with 80 bytes, so the sequence number of the next segment from the same host is 127+80 = 207. Assuming the TCP connection supports pipelining, the source port number for the second segment is the same as the first segment's source port number: 302. Similarly, with the same assumptions, the destination ports will be the same across both segments: 80

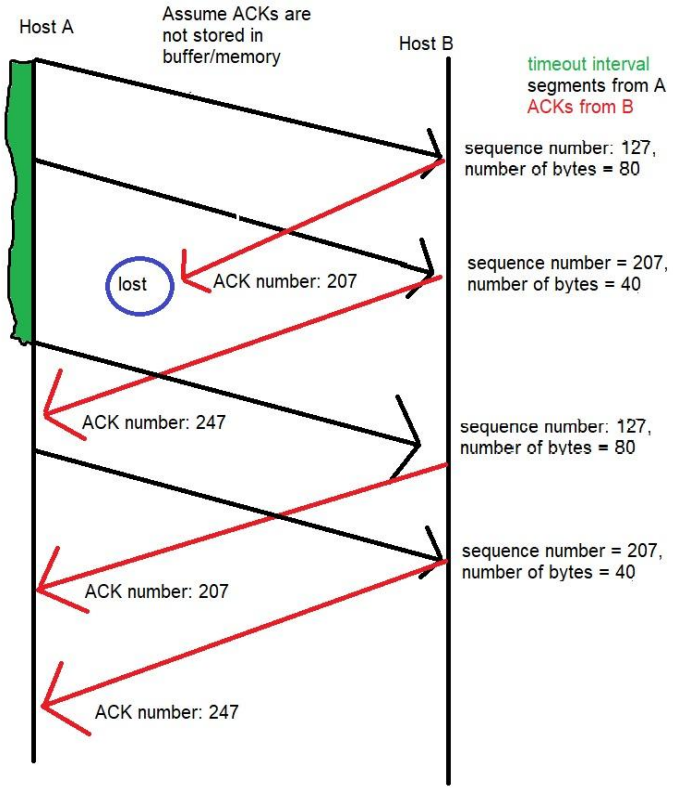
B.

In the ACK of the first segment, the ACK number is sum of the first segment's sequence number and the number of bytes in the first segment, so the ACK number of the first segment is 207, the source port is the first segment's destination port number: 80, and the ACK destination port is the first segment's source port: 302.

C.

In the ACK of the first arriving segment, the ACK number will be 127. This is because host B was expecting bytes 127 onwards, but it has not received the expected bytes yet, so it sends an ACK with the number of the next byte it expects. This is how host A knows that the first segment was not received.

D.



37.

A.

GBN: Sends 1-5, then resends 2-5, meaning GBN sends 5 + 4 = 9 segments before all segments are received, receiver sends 8 ACKs, 1 for each segment received. 9 sent segments - 1 lost segment = 8 ACKs. Sent sequence numbers: 1, 2, 3, 4, 5, then 2, 3, 4, 5. Sent ACK numbers: 1, 1, 1, 1, then 2, 3, 4, 5

SR: Sends 1-5, then only resends 2, meaning SR sends 5 + 1 = 6 segments before all segments are received, receiver sends 5 ACKs, 1 for each segment received. 6 sent segments - 1 lost segment = 5 ACKs. Sent Sequence numbers: 1, 2, 3, 4, 5, then 2. Sent ACK numbers: 1, 3, 4, 5, then 2

TCP: Assuming pipelining, Sends 1-5, then resends 2, meaning TCP sends 5 + 1 = 6 segments before all segments are received, receiver sends 5 ACKs, 1 for each segment received. 6 sent segments - 1 lost segment = 5 ACKs. Sent sequence numbers: 1, 2, 3, 4, 5, then 2. Sent ACK numbers: 2, 2, 2, 2, 6 since for TCP, ACK number is the number of the next expected sequence number, so the 6 means that it has received all segments 1-5.

B.

TCP would give the best response time due to TCP always resending segments upon arrival of an unexpected ACK number. SR and GBN have to wait for a timeout to occur before they resend packets.

40.

A.

Slow start is occurring between round intervals [1,6] and [23, 26] because it occurs when congestion is very low, so congestion avoidance would not make sense for these time intervals. Window size doubles, and then switches to congestion avoidance

B.

Congestion avoidance occurs during the round interval [6, 23] because it follows a linear path up until loss occurs, then the congestion window size gets cut in half, which is the distinguishing behavior of congestion avoidance.

C.

The loss event is triggered by duplicate ACKs, since the window size did not decrease all the way, just by about half.

D.

The loss event was triggered by a timeout, which is evidenced by the fact that the window size dropped all the way down to 1.

E.

The first round has ssthresh at 32, because this is the value at which congestion avoidance started.

F.

The ssthresh value from at 18th transmission round is 21, which is half of the congestion window size at the specified round.

G.

The value of ssthresh at the 24th transmission round is 13 because it is half (rounded up) of the congestion window size at the 24th round.

H.

The 70th segment is sent during round 7 because only 63 segments are sent during rounds 1-6 cumulatively, and 33 segments are sent suring round 7 (total of 96), so the 70th segment must have been sent during this round.

I.

Ssthresh will be 4 (half of the current window size of 8), and window size will be 4 + 3 MSS, which is 7.

J.

Window size will be 1 because Tahoe does not use fast recovery, so window size starts at 1 with a halved window size as ssthresh every loss. Window size was 42, so ssthresh is now 21

K.

1+2+4+8+16+21 = 52 segments sent between rounds 17 and 22, inclusive (reaches ssthresh of 21 on round 22)

44.

A.

1RTT -> 7MSS; 2RTT -> 8MSS; 3RTT -> 9MSS; 4RTT -> 10MSS; 5RTT -> 11MSS; 6RTT -> 12MSS

B.

At time t = 6RTT, cwnd = 12 MSS, but a batch of 12 MSS has not been sent, since from t = 5RTT to t = 6RTT, 11 MSS is sent (basis for increasing cwnd), So from t = 0 to t = 6RTT, batches sent = [6MSS, 7MSS, 8MSS, 9MSS, 10MSS, 11MSS]. Avg throughput = (6MSS + 7MSS + 8MSS + 9MSS + 10MSS + 11MSS)/6RTT = (13MSS + 17MSS + 21MSS)/6RTT = 51MSS/6RTT = 8.5MSS/RTT