Nhat Doan - CSC 138 ( T-TH 12:00 - 1:15)

***Homework 5- Exercise 2***

Ex2 (5 points)

1. Computing an Internet checksum

Consider the two 16-bit words (shown in binary) below. Recall that to compute the Internet checksum of a set of 16-bit words, we compute the one's complement sum [1] of the two words. That is, we add the two numbers together, making sure that any carry into the 17th bit of this initial sum is added back into the 1's place of the resulting sum); we then take the one's complement of the result. Compute the Internet checksum value for these two 16-bit words:

01100110 10011111 this binary number is 26271 decimal (base 10)

10101010 11010001 this binary number is 43729 decimal (base 10)

answer:

01100110 10011111

* 10101010 11010001

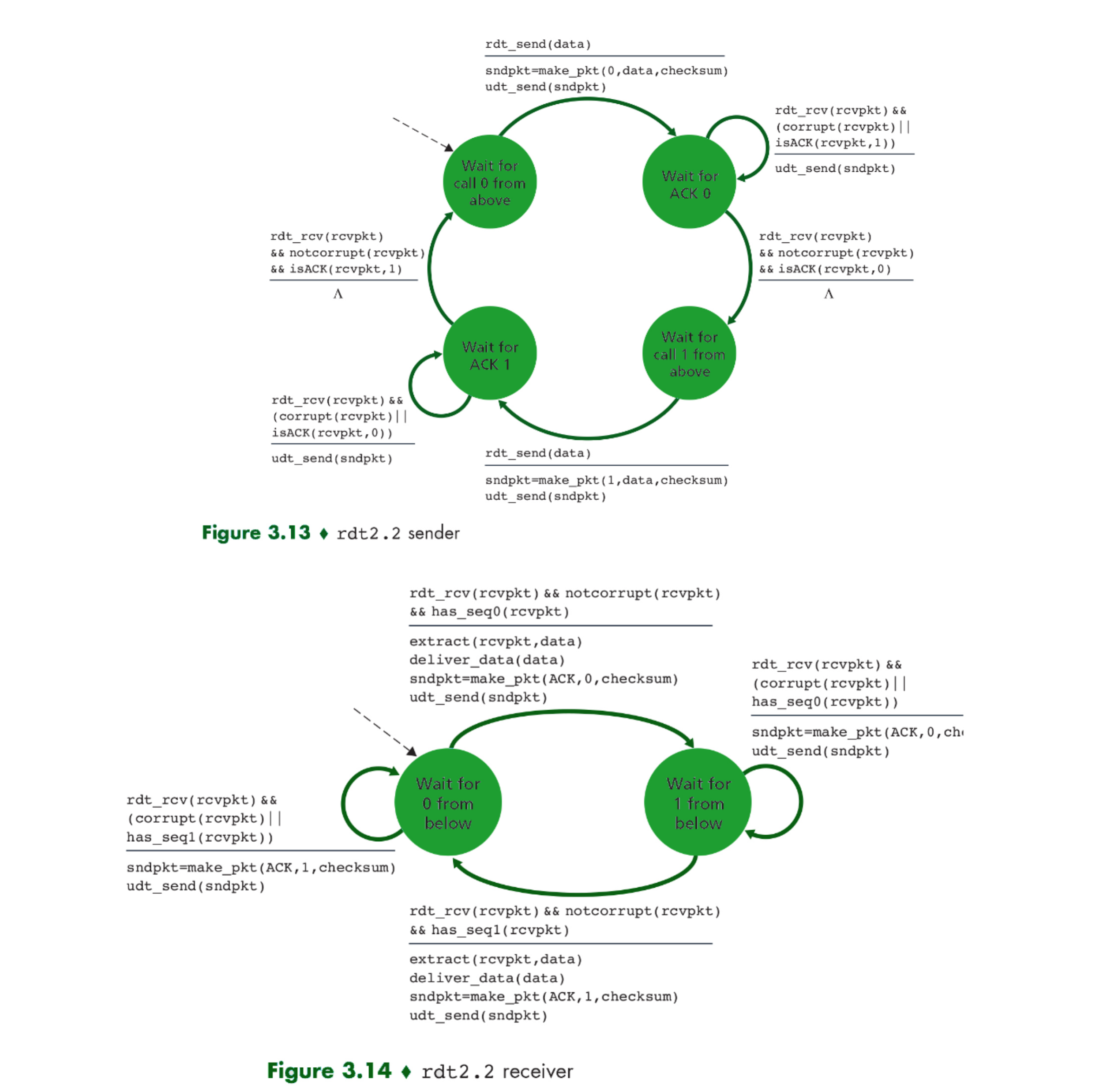
-----------------------------

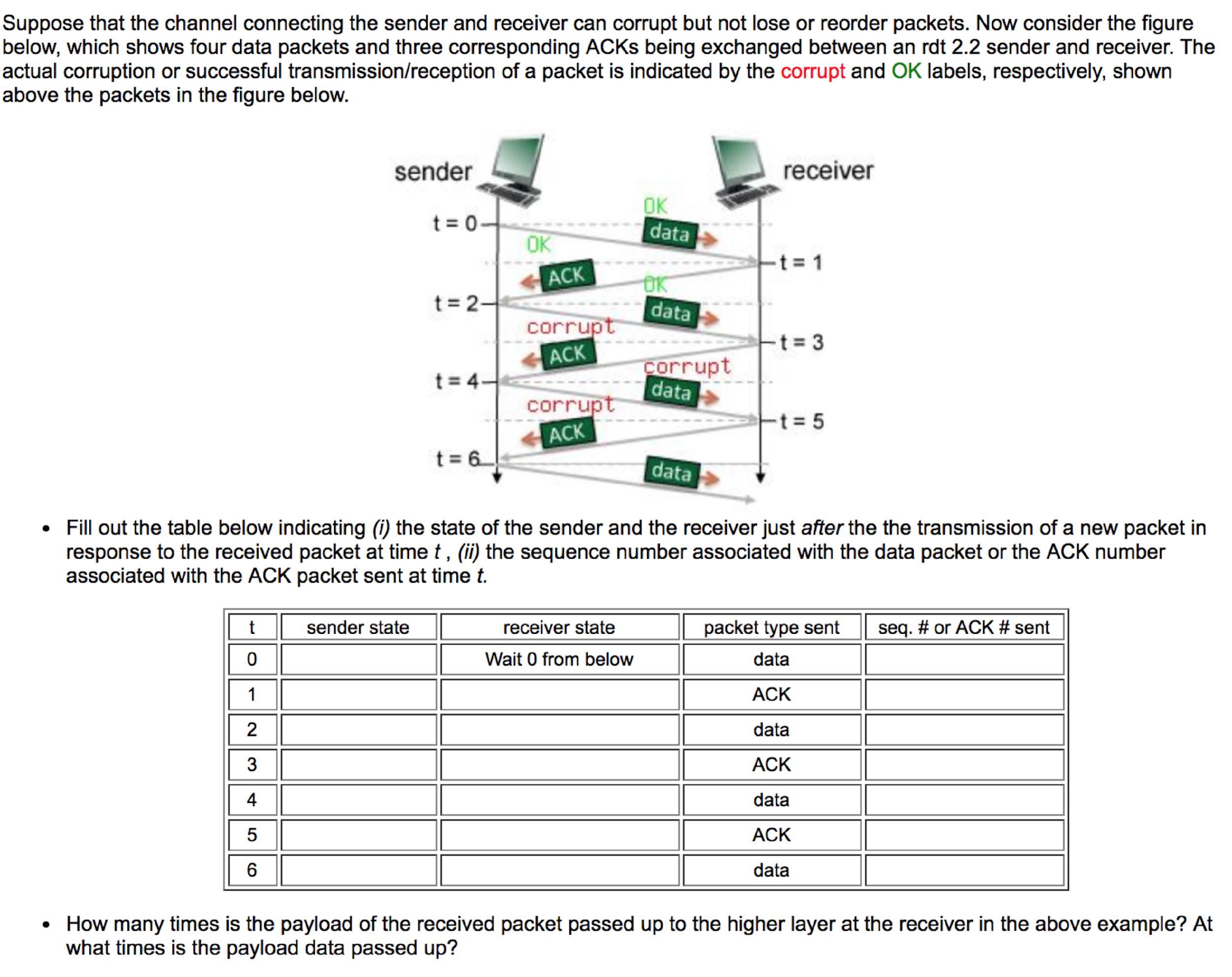
1 00010001 01110000

+ 1

1 complement **11101110 10001111**

2. Consider the rdt2.2 protocol from the text (pages 209-212). The sender and receiver FSMs for the sender and receiver are shown below:





Answer:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t | sender\_state | receiver state | packet type sent | seq # of ACK # sent |
| 0 | wait ACK 0 | Wait0 from below | data | 0 |
| 1 | wait ACK 0 | Wait1 from below | ACK | 0 |
| 2 | wait ACK 1 | Wait1 from below | data | 1 |
| 3 | wait ACK 1 | Wait0 from below | ACK | 1 |
| 4 | wait ACK 1 | Wait0 from below | data | 1 |
| 5 | wait ACK 1 | Wait0 from below | ACK | 1 |
| 6 | wait ACK 1 | Wait0 from below | data | 1 |

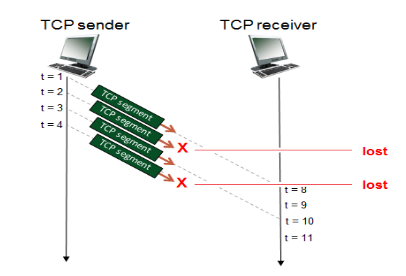
3.

Consider the figure below in which TCP a sender and receiver communicate over a connection in which the sender-to-receiver segments may be lost. The TCP sender sends initial window of four segments at t=1,2,3,4, respectively. Suppose the initial value of the sender-to-receiver sequence number is 149 and the first four segments each contain 556 bytes. The delay between the sender and the receiver is 7 time units, and so the first segment arrives at the receiver at t=8. As shown in the figure, two of the four segment(s) are lost between the sender and the receiver.

Answer the following questions:

Give the sequence numbers associated with each of the five segments sent by the sender

List the sequence of acknowledgements transmitted by the TCP receiver in response to the receipt of the segments actually received. In particular, give the value in the acknowledgement field of each receiver-to-sender acknowledgement, and give a brief explanation as to why that particular acknowledgement number value is being used



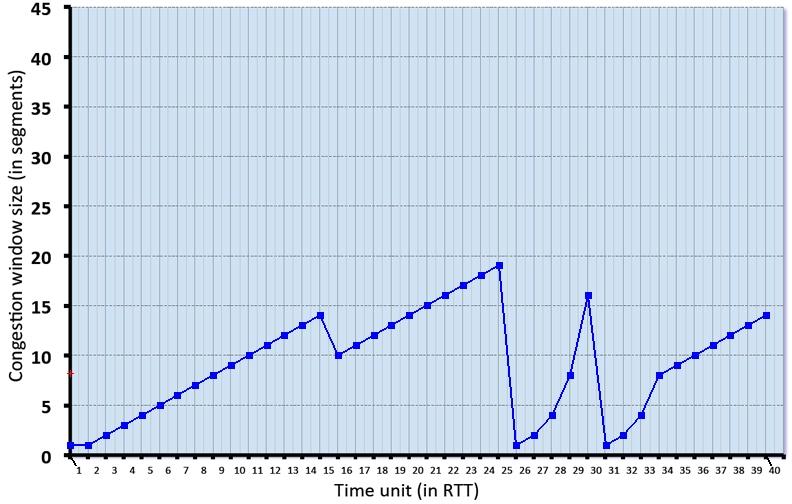
Answer:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sender-to-receiver | time segment sent | sender-to-receiver segment sequence number field value | time segment received, and ACK segment sent | receiver-to-sender ACK field value |
| Segment 1 | 1 | 149 | 8 | 705 |
| Segment 2 | 2 | 705 |  | No ACK sent ← segment was lost |
| Segment 3 | 3 | 1261 | 10 | 705. ACK this re-acknowledge the last correctly received, in-order byte(705) |
| Segment 4 | 4 | 1817 |  | No ACK sent ← segment was los |

4.

**TCP in action: slow start, congestion avoidance, and fast retransmit.**

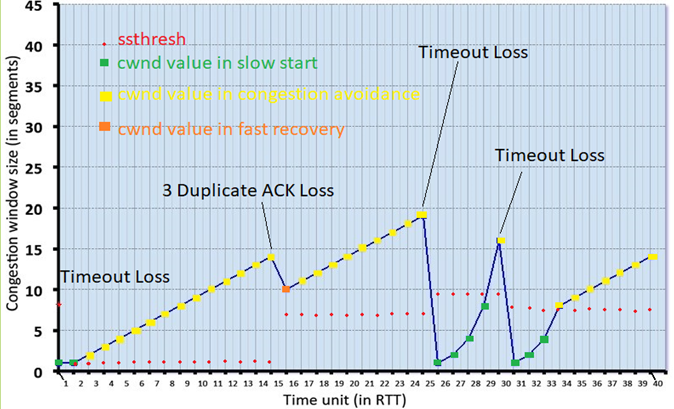
Consider the figure below, which plots the evolution of TCP's congestion window at the beginning of each time unit (where the unit of time is equal to the RTT); see Figure 3.53 in the text. In the abstract model for this problem, TCP sends a "flight" of packets of size cwnd at the beginning of each time unit. The result of sending that flight of packets is that either (i) all packets are ACKed at the end of the time unit, (ii) there is a timeout for the first packet, or (iii) there is a triple duplicate ACK for the first packet. In this problem, you are asked to reconstruct the sequence of events (ACKs, losses) that resulted in the evolution of TCP's cwnd shown below.



Consider the evolution of TCP's congestion window in the example above and answer the following questions. The initial value of cwnd is 1 and the initial value of ssthresh (shown as a red +) is 8.

* Give the times at which TCP is in slow start, congestion avoidance and fast recovery at the start of a time slot, when the flight of packets is sent.
* Give the times at which the first packet in the sent flight of packets is lost, and indicate whether that packet loss is detected via timeout, or by triple duplicate ACKs.
* Give the times at which the value of ssthresh changes, and give the new value of ssthresh

Answer:

* 

Time in slow start (in RTT): 1-2, 26-29, 31-33

Time in congestion avoidance (in RTT): 3-15, 17-25, 30, 34-40

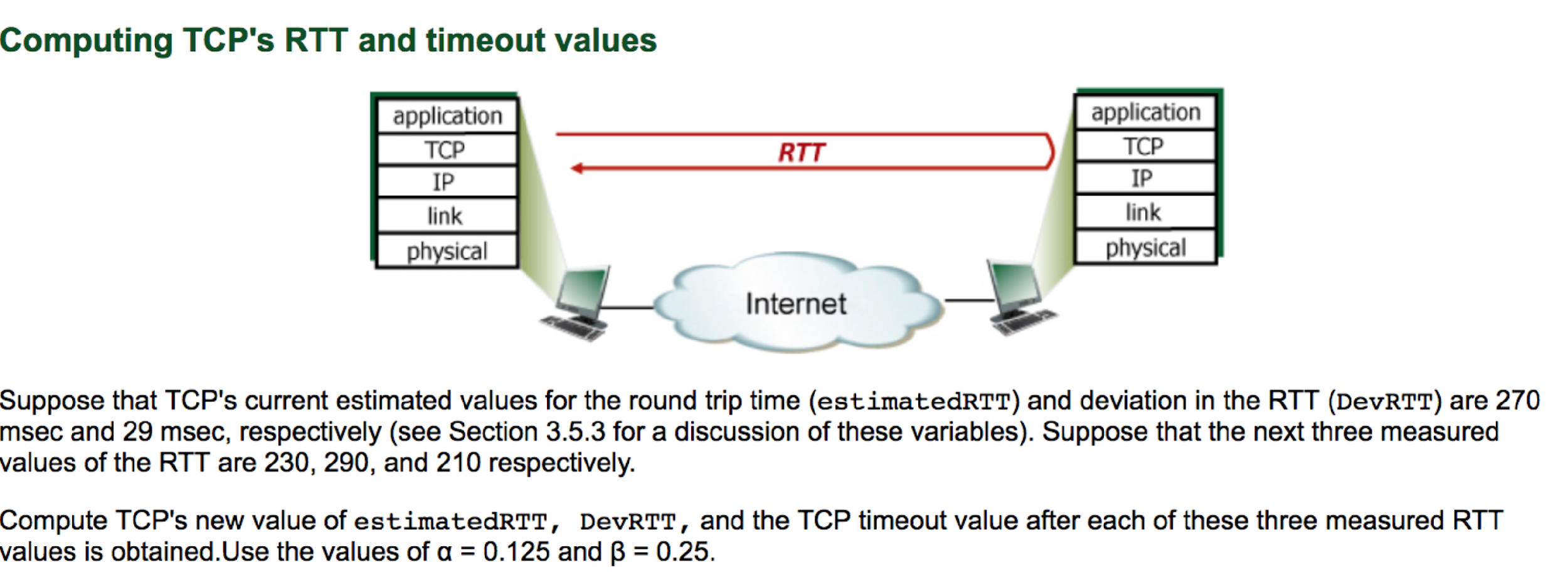
Time in fast recovery (in RTT): 15

slow start( RTT) : 1-2, 26-29, 31-33

First package is lost at 1 RTT as Timeout Loss

|  |  |
| --- | --- |
| Time SSthresh changes (in RTT) | New ssthresh values ( in segment) |
| 1 | 1 |
| 15 | 7 |
| 25 | 9.5 |
| 30 | 8 |

5.



**Answer:**

After the first RTT :

estimatedRTT = 0.875\*270 + 0.125\*230 = 265 msecs

DevRTT = 0.75\*29 + 0.25\*(abs(230-265)) = 30.5 msecs

TimeoutInterval = 265 + 4\*30.5 = 387 msecs

After the second RTT:

estimatedRTT = 0.875\*265 + 0.125\*290 = 268.125 msecs

DevRTT = 0.75\*30.5 + 0.25\*(abs(290 – 268.125)) = 28.34375 msecs

TimeoutInterval = 268.125 + 4\*28.34375 = 381.5 msecs

After the third RTT:

estimatedRTT = 0.875\*268.125 + 0.125\*210 = 260.859375 msecs

DevRTT = 0.75\*28.34375 + 0.25\*(abs(210 - 260.859375)) = 33.97265625 msecs

TimeoutInterval = 260.859375 + 4\*33.97265625 = 396.75 msecs