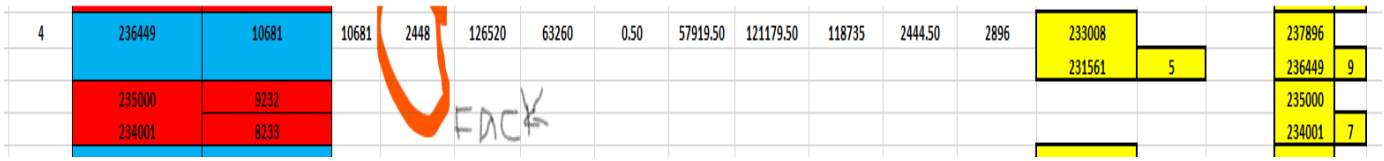
NAME: **ALI SURA OZDEMIR** **PID: 0933720**

ECE 6321 - TAKE HOME EXAM - 2

1. *Use Sender's Wireshark data to verify the byte range in the highest SACK field reported by the 4th duplicate ACK during the recovery period. Verify the byte ranges of the missing segments reported by this duplicate ACK.*

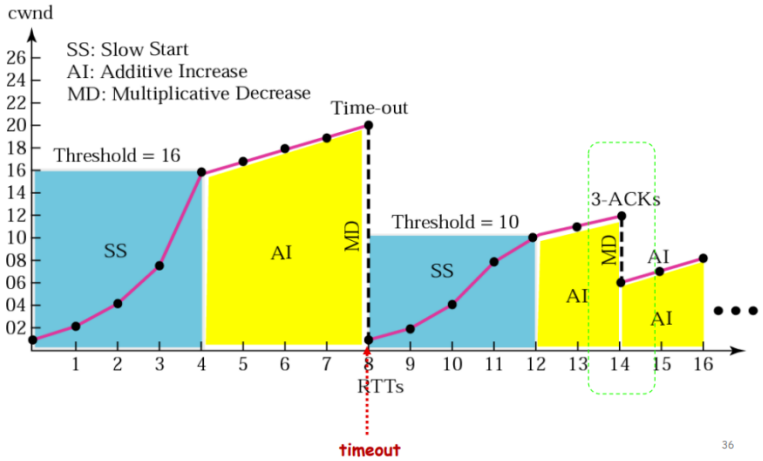
Whenever an out-of-order segment is received, duplicate ACK is sent back to the sender. The forward ack (FACK) of a duplicate ack is 1 less than the highest byte number of its highest SACK block. From the Scoreboard" data file we can see that forward acknowledgement number or FACK is "2448". Byte range of lost segment reported by 4th duplicate ACK on scoreboard is given as below:



Below is the Wireshark we observe the transmission loss:

2. *We have shown that the two parameters Cwind and Cwindmultiplier determine retransmission timing during the recovery period. Carry out a series of Scoreboard experiments to determine the effect of each parameter on transmission timing. Discuss your conclusions.*

Congestion Avoidance: Additive Increase : To avoid congestion before it happens, we must slow down this exponential growth. TCP defines another algorithm called congestion avoidance, which increases the cwnd additively instead of exponentially. When the size of the congestion window reaches the slow-start threshold in the case where cwnd = i, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole "window" of segments is acknowledged, the size of the congestion window is increased by one. A window is the number of segments transmitted during RTT.

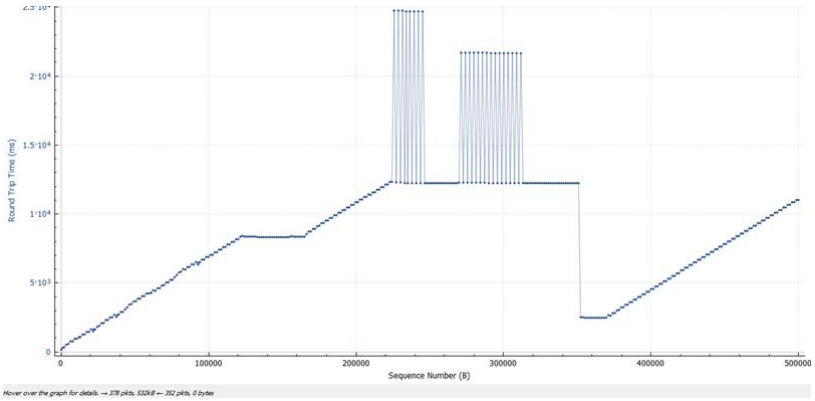


The sender starts with cwnd = 4. This means that the sender can send only four segments. After four ACKs arrive, the acknowledged segments are purged from the window, which means there is now one extra empty segment slot in the window. The size of the congestion window is also increased by 1. The size of window is now 5. After sending five segments and receiving five acknowledgments for them, the size of the congestion window now becomes 6, and so on. In other words, the size of the congestion window in this algorithm is also a function of the number of ACKs that have arrived and can be determined as follows: If an ACK arrives, cwnd 5 cwnd 1 (1/cwnd). The size of the window increases only 1/cwnd portion of MSS (in bytes). In other words, all segments in the previous window should be acknowledged to increase the window 1 MSS bytes. If we look at the size of the cwnd in terms of round-trip times (RTTs), we find that the growth rate is linear in terms of each round-trip time, which is much more conservative than the slow-start approach.

Start	$\rightarrow$	$cwnd = i$
After 1 RTT	$\rightarrow$	$cwnd = i + 1$
After 2 RTT	$\rightarrow$	$cwnd = i + 2$
After 3 RTT	$\rightarrow$	$cwnd = i + 3$

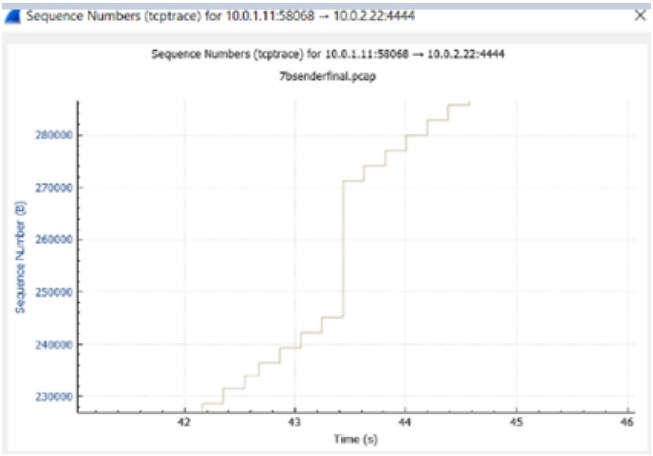
In the congestion-avoidance algorithm, the size of the congestion window increases additively until congestion is detected.

Congestion Detection: Multiplicative Decrease: If congestion occurs, the congestion window size must be decreased. The only way the sender can guess that congestion has occurred is by the need to retransmit a segment. However, retransmission can occur in one of two cases: when a timer times out or when three Duplicate ACKs are received. In both cases, the size of the threshold is dropped to one-half, a multiplicative decrease.



As in the Wireshark graph, because of the Duplicate ACK, Congestion Multiplication Decrease occurs. Less possible to have a congestion when three Duplicate ACKs received, segment drops; Congestion Avoidance phase starts while **Cwnd** sets the threshold value. As a result of **C_multiplicative decrease**, the RTT drops rapidly and then rises when the real packet is sent from the sender to the receiver.

3. *Frame 514 of the senders Wireshark file is the last duplicate ACK of the recovery period. What follows is a series of cumulative ACKs triggered by the arrival of retransmitted segments at the receiver. These ACKs return new values of the receiver's advertised window. Starting from 127,448 in frame 514, the advertised window decreases to a minimum value of 75,268, whereupon it begins to increase again. Explain this behavior.*



Here, the scenario is a transfer of control from CWIN to RWIN. TCP sends a "flight" of packets of size ***cwnd*** at the beginning of each time unit. When the last LOST segment is received (initially), RWIN controls – and big ACK is sent. However, last byte of the segment must be in the advertised window. Since, this can only go up to a maximum advertised window (RWND). Then the process gets back to slow start again and transmission happens rapidly.

In slow start congestion control, TCP increases the window's size rapidly to reach the maximum transfer rate as fast as possible. This self-imposed window size increases as TCP confirms the network's ability to transmit the data without errors. However, this can only go up to a maximum advertised window (RWND). The RWIN dictates how much data a TCP receiver is willing to accept before sending an acknowledgement (ACK) back to the TCP sender. The receiver will advertise its RWIN to the sender and thus the sender knows it can't send more than an RWINs worth of data before receiving and ACK from receiver. The SWIN dictates how much data a TCP sender will be allowed send before it must receive an ACK from the TCP receiver. The CWIN is a variable that changes dynamically according to the conditions of the network. If data is lost or delivered out-of-order the CWIN is typically reduced.