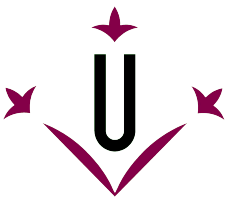
103087 - ICT Project Communication Services and Security

Activity 1

Jordi Rafael Lazo Florensa
Alejandro Clavera Poza

19 March 2023

Master's degree in Computer Engineering



Universitat de Lleida
Escola Politècnica Superior

1 Problem 1

1. **Probability of a segment being dropped with a AvgLen=8**.

   To calculate P we base ourselves on the y, since it is in the interval (MinTh-MaxTh) this probability can be represented in the following expression:

   $$P = MaxP * \frac{AvgLen - MinTh}{MaxTh - MinTh} = 0,4 * \frac{8-4}{10-4} = 0,4 * \frac{2}{3} = 0,26$$

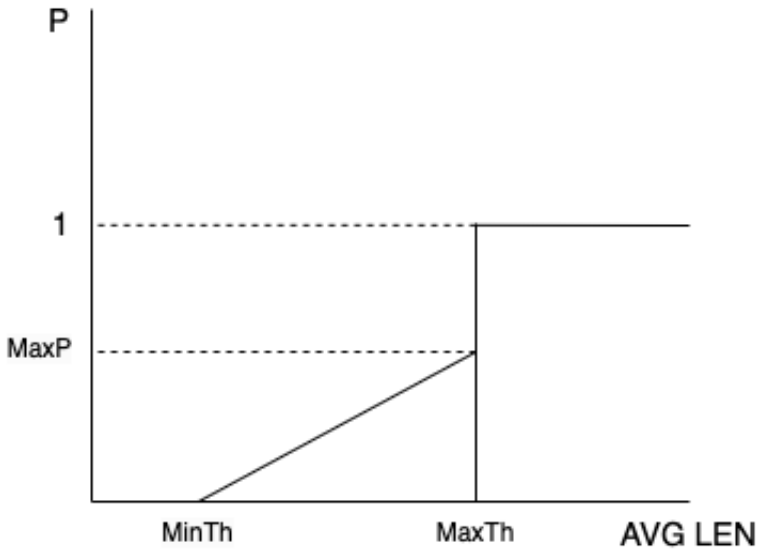


Figure 1: Graphical representation of a dropped segment

2. **Probability that 3 consecutive segments enter into the queue, assuming that all of them find the same average queue length (AvgLen) of 8**.

   Because all 3 segments have the same AvgLen, they will also have the same probability of being removed.

   $P = 0,26 \rightarrow P_e = 1 - P = 0,74$

   Therefore the probability that all 3 segments will enter the queue is:

   $P = P_{e1} * P_{e2} * P_{e3} = 0,74 \cdot 0,74 \cdot 0,74 = 0,40$

3. **Same probability as previous point (2) assuming a modified RED congestion control where the probability of a segment being dropped is $\frac{P}{1 compt \cdot P}$, being P the same probability computed at point (1). compt is the number of segments that entered into the queue from the last dropped segment. Assume compt = 0 for the first segment entering into the queue**.

   For this case, despite the fact that the 3 segments have the same AvgLen=8, the modified version of RED is being applied, which takes into account the number of queued segments before a drop occurs. Therefore, the probabilities that 1 segment has been eliminated is different, being necessary to calculate them independently under the following expression:

$P = \frac{P'}{1-compt \cdot P'}$ $\quad P' = 0,26$

Calculation of the probabilities of being eliminated:

$P_1 = \frac{0,26}{1-0 \cdot 0,26} = 0,26 \quad P_2 = \frac{0,26}{1-1 \cdot 0,26} = 0,35 \quad P_3 = \frac{0,26}{1-2 \cdot 0,26} = 0,54$

Calculation of the probabilities of entering the queue:

$P_{e1} = 1 - P_1 = 0,74 \quad P_{e2} = 1 - P_2 = 0,65 \quad P_{e3} = 1 - P_3 = 0,46$

Calculation of the probability that 3 consecutive segments enter the queue:

$P_E = P_{e1} \cdot P_{e2} \cdot P_{e3} = 0,74 \cdot 0.65 \cdot 0,46 = 0,22$

2 Problem 3

1. **What two regular expressions allow to obtain:**

   • Segments sent from TCP agent.

   ```
   ^-(?!.*ack).*
   ```

   • Acks received at TCP agent.

   ```
   ^r \S+ \S+ 1 .*ack.*
   ```

   Where:

   ```
   \S = [^ \t\n\r\f\v]
   ```

2. **Using three different TCP agents:**

   • RFC793 with original congestion control.

   • RFC793 with slow start.

   • Reno.

3. **Show an example obtained from your simulation where fast retransmission is produced.**

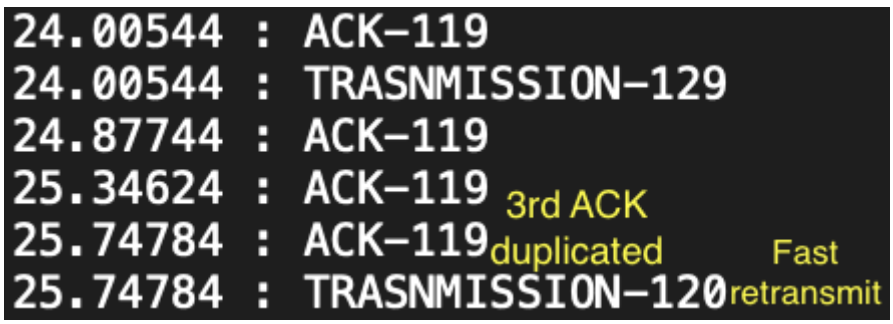


Figure 2: Moment when fast retransmit starts in the trace

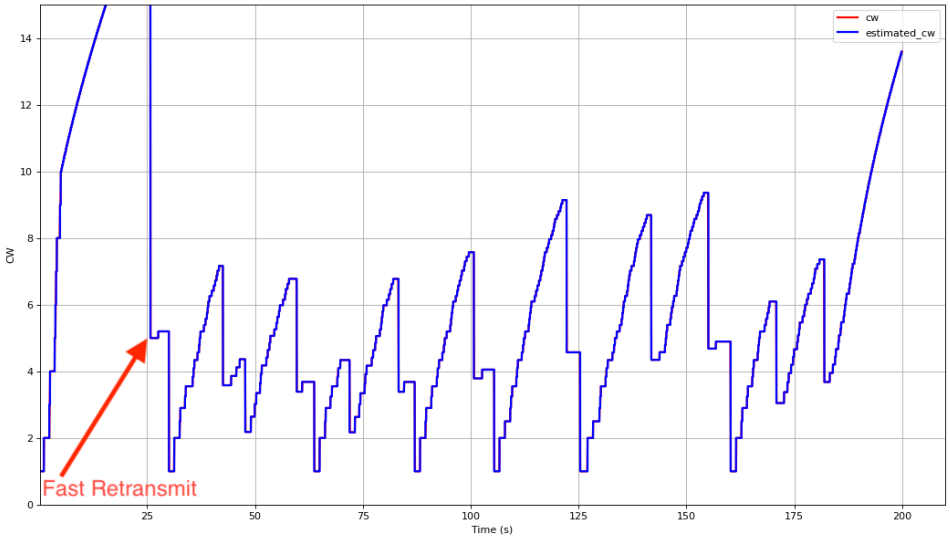


Figure 3: Moment when fast retransmit starts

In this image you can see how in the second approximately 25 the fast retransmit starts after receiving the third duplicate ack.

4. **For each one of the three agents, explain which is the policy for retransmissions when the congestion window reaches its limit and can no longer budge.**

   For the RFC793 agent with original congestion control, the retransmission policy consists of waiting for the timeout of the current congestion window to occur, after which it will try to retransmit all the lost segments of that window, for this the first one will be retransmitted initially and it will wait to receive the ack from it (since when the timeout occurs the congestion window has been reduced to its minimum value). Once the ack has been received, the agent will set the congestion window to the maximum and will retransmit the rest of the segments that fit in that window. This action will be repeated as subsequent ack segments arrive until all lost segments have been retransmitted.

   The transmission policy of the RFC 793 with slow start agent is based on the same idea as the RFC 793 with original congestion control agent commented in the previous paragraph. The retransmissions are made at the time the timeout occurs. From this point, in the same way, an attempt will be made to retransmit all the segments with the difference that this action is less aggressive since when applying the slow start process the congestion window grows progressively so that the retransmission network does not collapse.

   Finally, the Reno agent's transmission policy consists of anticipating a timeout to occur in order to reduce downtime. To do this, once the third repeated ack is received, it enters a fast recovery phase in which it retransmits the first segment that has been lost (during this action, the congestion window is reduced by half). Once this segment is retransmitted, the agent, if the window allows it, will continue

retransmitting more segments. Once the ack is received for the first lost segment, it will exit the fast recovery phase and continue acting normally (linear increment). Otherwise, if the ack is not received and before the timeout is triggered, the agent will reduce the congestion window to a minimum and begin a new slow start phase.

5. **Run a fourth simulation with TCP NewReno agent. Measure throughput and number of retranmissions from the simulation trace and compare it against TCP Reno.**

   - Retransmission in TCP Reno: 28
   - Retransmission in TCP NewReno: 47

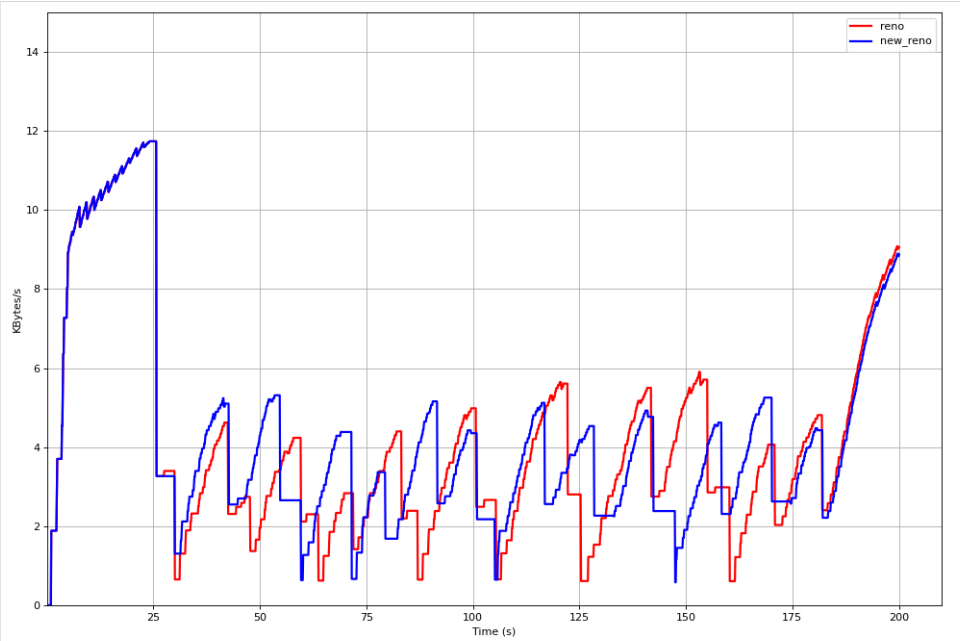


Figure 4: Throughput of TCP Reno vs TCP New reno

As can be seen, the number of retransmissions and the Throughput of the TCP NewReno agent in general is greater than the TCP Reno agent due to the improvement of the partial ACKs and to the reduction of the number of duplicate ACKs. By reducing this number the TCP agent newReno gives up the segments much sooner, so it speeds up retransmissions.

6. **For TCP New Reno show an example obtained from your simulation where partial Acks were observed.**

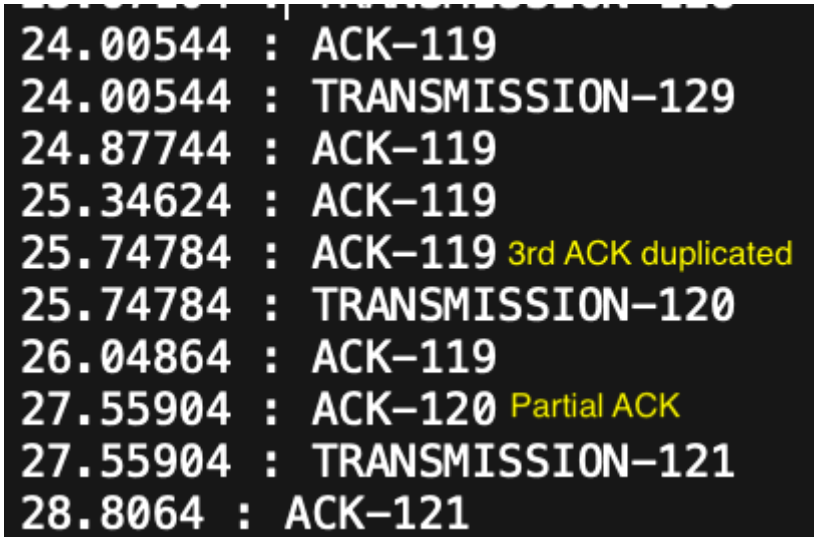


Figure 5: Partial ACK

As we can see in the capture of the trace, the agent started the fast retransmission when the third ack arrived at second 25.74. At this point the agent retransmitted the lost segment 120. A short time later, another duplicate ack arrived (specifically 120), at this point is when the partial ack comes into play. Instead of waiting for two more duplicates, the agent retransmits the next lost segment of the window (segment 121). Thanks to this, the TCP new reno agent can recover more than one packet before the timeout occurs.

7. **Run a fifth simulation using TCP Reno but, in this case, use RED at n2 with MinTh=10 and MaxTh= 20. Compare the throughput and congestion window plots with TCP Reno without RED previous simulation.**

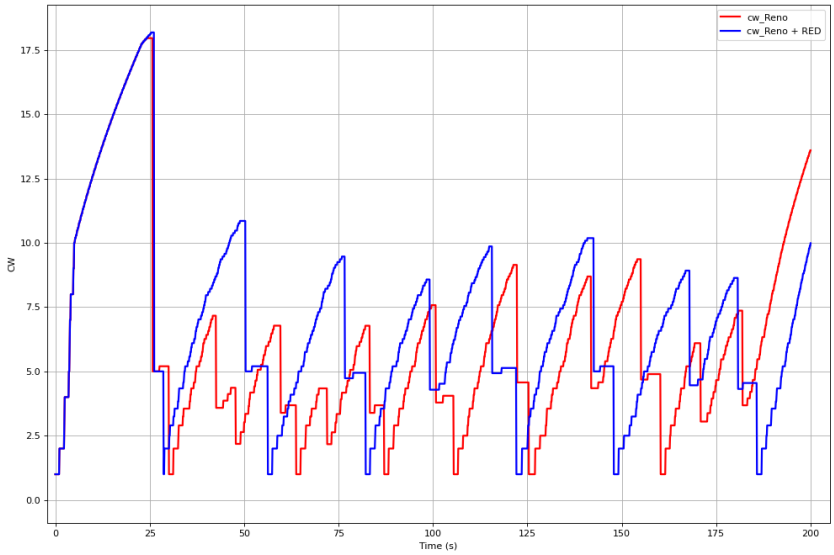


Figure 6: Congestion window of TCP Reno vs TCP Reno + RED

As can be seen in the previous Figure, the evolution of the congestion window throughout the simulation when applying RED improves with respect to not applying it. The main reason for this is the difference between the RED application and the previously used method which only discards segments when the queues reach their top (discarding both incoming segments and elements within the queue arbitrarily).

If we analyze the trace of the simulation, it can be clearly seen that most of the congestion is produced by the UDP traffic coming from node 0. By applying RED, it is possible to reduce the number of TCP packet losses based on discarding UDP traffic automatically. sporadic which has a high probability of being eliminated when dealing with majority traffic.

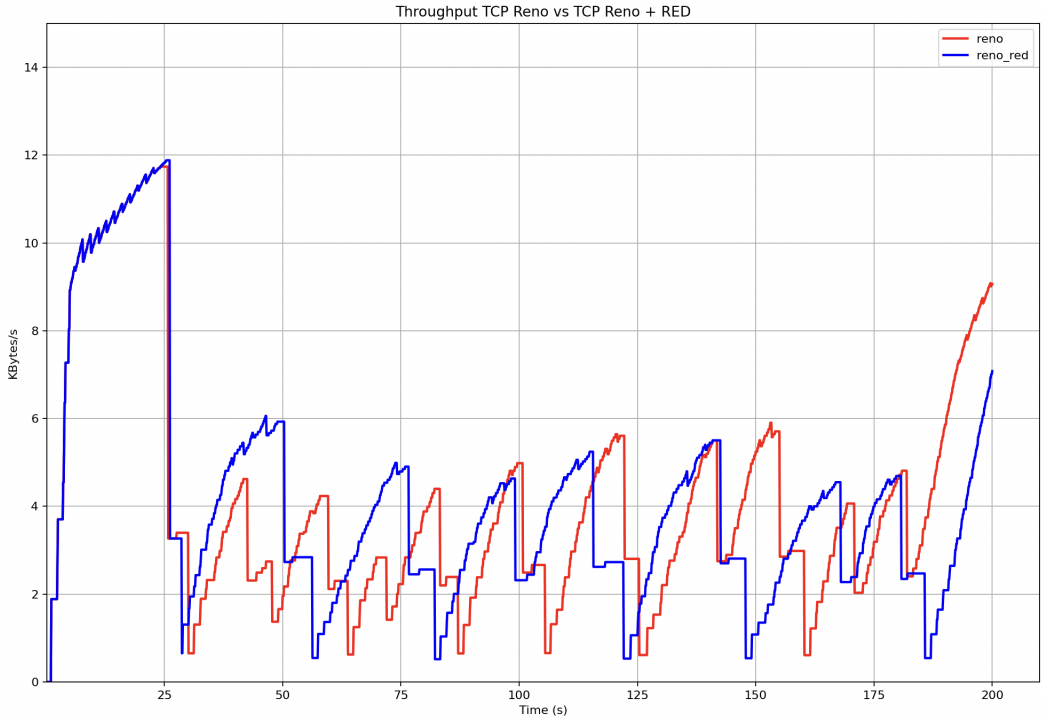


Figure 7: Throughput of TCP Reno vs TCP Reno + RED

On the other hand, if we compare the performance we can see that both cases have a very similar value. This is because both simulations work on the same agent, so the behavior in a congestion situation is the same. Despite the fact that the packet discard mechanisms of the nodes in both simulations are different, congestion situations continue to occur at approximately the same time intervals, so the agents will perform very similar actions among themselves and thus describe a throughput. similar where it seems the New Reno agent manages to stick around for longer periods.