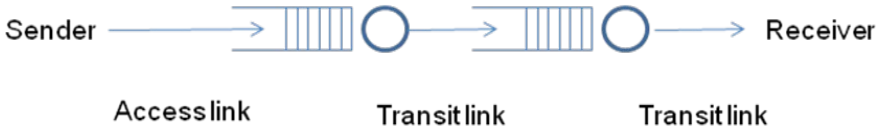
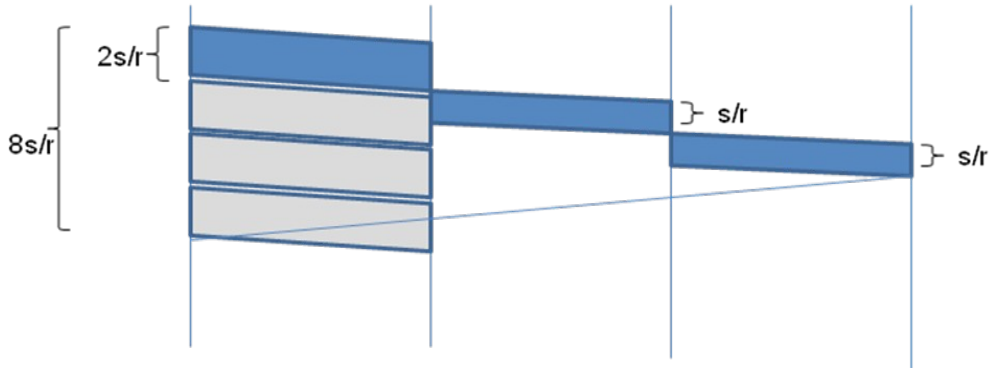
# COL334/672: Assignment 4

1. We have the following topology: a sender communicating to a receiver via a series of two routers. Packets are of size s, the transit links have a transmission rate of r, while the access link operates at half the rate of the transit links. The round trip propagation delay is 4s/r, hence within an RTT of 8s/r the access link can just about support a window size of 4 packets. Ignore acknowledgement sizes.



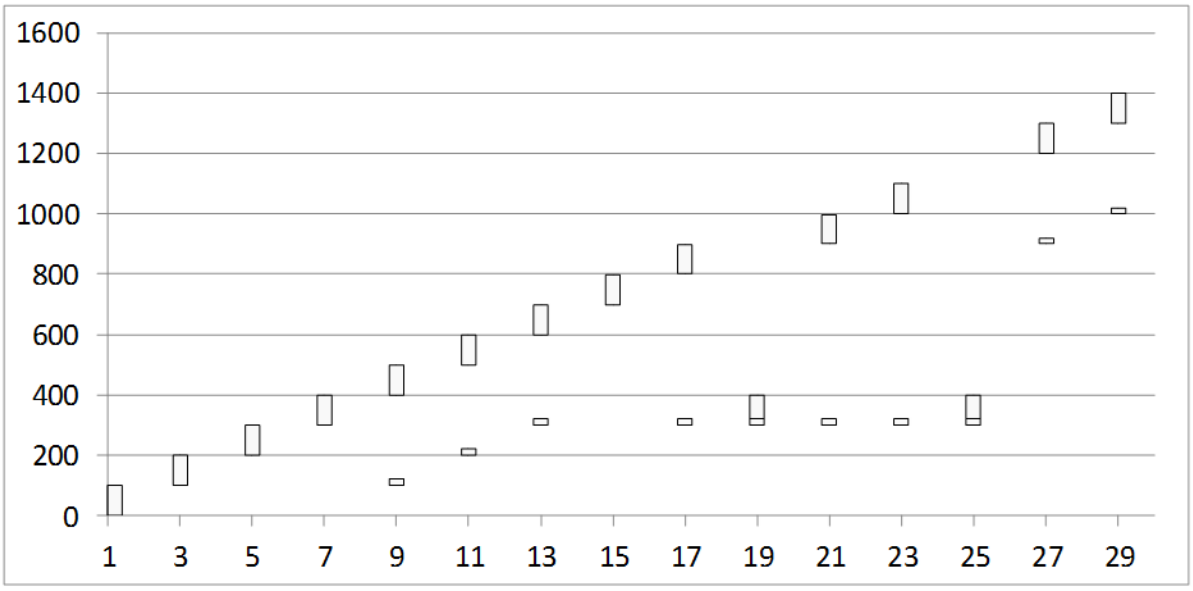

Consider the following packet trace at the sender using a transport protocol similar to TCP. The y-axis indicates sequence numbers in bytes – long vertical rectangles are packets and each packet is 100 bytes long, thus the first packet contains data from sequence number 0 to 99, the second packet from sequence number 100 to 199, etc. The x-axis indicates time in units of s/r. The small stubs are acknowledgement numbers – thus, the stub at time 9 (after one RTT) is the cumulative acknowledgement for the first packet with sequence number 0 to 99, the stub at time unit 11 is the cumulative acknowledgement for the second packet, etc.

The congestion window indicates the maximum number of unacked packets that can be sent. Assume this window size to be fixed and much greater than 4 packets – this implies that the sender will try to push out a packet every 2 time units, which is the maximum transmission rate its access link allows.

Also assume for simplicity that no acknowledgements are lost and no reordering occurs.

Fast retransmission is triggered upon getting triple duplicate acks, taking into account the very first ack as well.

Now explain the packet trace below.




i. Packet 300-399 seems to have been lost. What triggers the retransmission of the lost packet?

   The triple duplicate ack (three acks at t = 13s/r, 17s/r, 19s/r) triggers the immediate retransmission of the lost packet.

ii. The acknowledgement received at time 21 was generated at the receipt of which packet at the receiver?

   It was generated at the receipt of the packet sent at time t = 13s/r (which has bytes 600-699).

iii. Why is the acknowledgement at time 21 still referring to the lost packet even though it has been retransmitted?

   The packet won't reach the receiver till before 2s/r time has elapsed, and the acknowledgement doesn't reach the sender till before 4s/r time. Hence the lost packet hasn't reached the receiver at time 21. Even though packets beyond the lost packet have been received, the acknowledgement refers to the lost packet because of the cumulative acknowledgement mechanism – till the lost packet is not received, it keeps sending the acknowledgement for the lost packet.

iv. Why is the lost packet again retransmitted at time 25?

   The last three acknowledgements refer to the lost packet (since it was still not received at the receiver's end), so it resends due to a triple duplicate ack.

v. Why is there a sudden jump in the acknowledgement number at time 27? The acknowledgement was generated at the receipt of which packet?

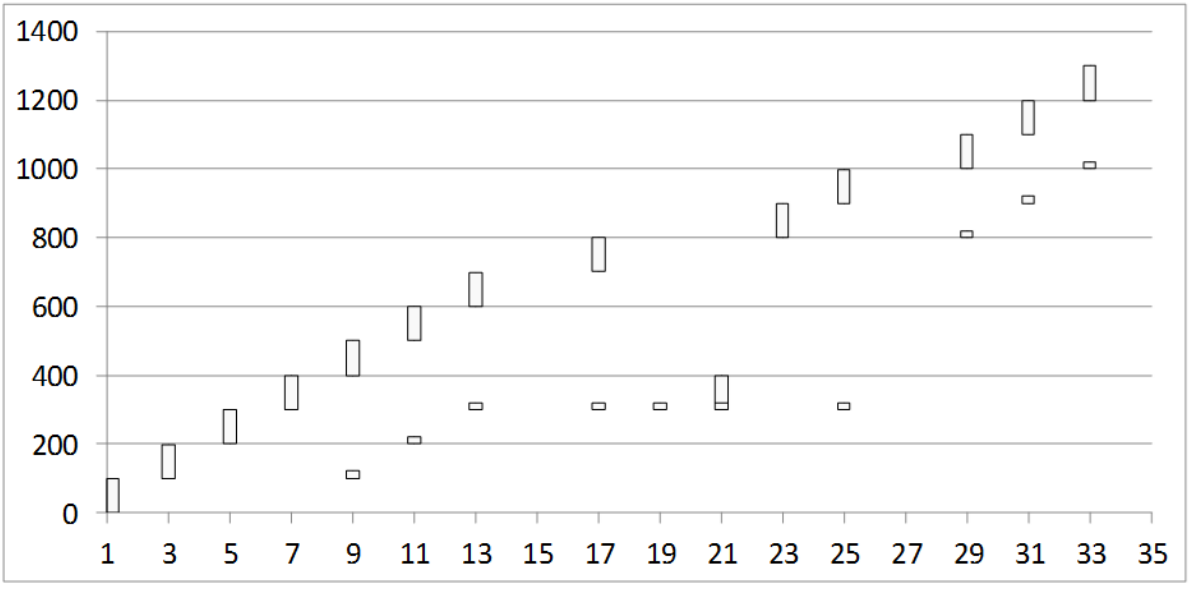
   This is because many packets (precisely the ones sent from times 9 to 17) after

the lost packet have been received which should have been acknowledged if there was no lost packet (since the acknowledgement is cumulative), so the cumulative ack is only till the lost packet. Once that packet is received, the receiver sends an acknowledgement, for all the remaining packets too. The acknowledgement was generated at the receipt of packet 300-399 sent at t = 19s/r due to the RTT delay of 4s/r (and not due to the one at 25s/r, which was sent because there is an RTT delay of 4s/r, which is more than the time for the triple duplicate acks, and hence it "slips" through before the ack triggered by the retransmission reaches the sender).

2. In another variant, the initial congestion window size is started at 4 packets, and the window is incremented fractionally by (1 / int(current window size)) upon receiving an acknowledgement. Thus, a window size of 4 will increment to 5 after having received 4 acknowledgements (4.25 after the first ack, 4.5 after the second ack, 4.75 after the third, and 5 after the fourth ack). Note that packets are dispatched only if they can be accommodated fully within the window, ie. even with a window of 4.75 only four outstanding packets will be allowed.

   The window size also reduces to half upon receiving triple duplicate acknowledgements which is seen as an evidence of packet loss. Note that receipt of the third dup ack will not increment the window, ie. if the window is 5 when the third dup ack arrives, it will just be reduced to 2.5, and not add another 1 / int(2.5) increment for this ack as is done for other acks. Also note that the event of a triple dup ack is also interpreted as a loss, and hence the number of outstanding packets will be assumed to be one less than what was it estimated to be earlier. This is almost identical to TCP operations in the congestion avoidance phase.

   Answer the following questions. Hint: Maintain two variables for congestion window and the outstanding data to understand what is happening.



i. What is the window size at time 17?

5 (it is the 4th ack, so the window size changes as 4 -> 4.25 -> 4.5 -> 4.75 -> 5 at each ack).

ii. What is the window size at time 19? Why is no packet pushed out at time 19?

2.5 since the triple duplicate ack arrives at time 19, so window size becomes halved from its previous value at the previous ack. Note that at this time, packet 0-99, 100-199, 200-299 have been sent and acknowledged, and packet 300-399 is lost. The remaining packets except the last two have been accounted for by the remaining acks, so there are exactly 2 packets in the window, whose size is 2.5. Hence there is no packet pushed out at time 19.

iii. What is the window size at time 21? What is the outstanding data estimated by the sender at time 21?

The window size just after time 21 is 3, which increases because of the ack (2.5 + 1/int(2.5) = 3). The outstanding data estimated by the sender just before the ack consists of 2 packets (600-699, 700-799), just after the ack consists of exactly one packet (700-799), and just after retransmission consists of 2 packets (700-799, 300-399).

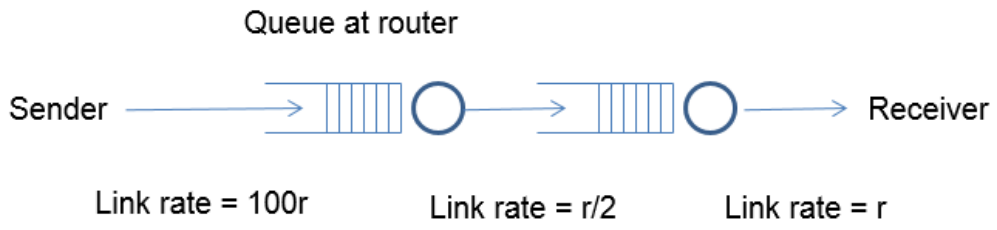
iv. Why is a packet pushed out at time 23 even though no ack is received at that time?

Note that just after time 21, the window size is 3, however there are exactly 2 packets in the window (after including the packet sent at time 21). So it has another packet that it can fit in the window, so it needs to send another packet. However, the sending of one packet takes time at least 2s/r, and thus the second packet gets transmitted at time 21+2, which is 23.

v. What is the window size at time 31?

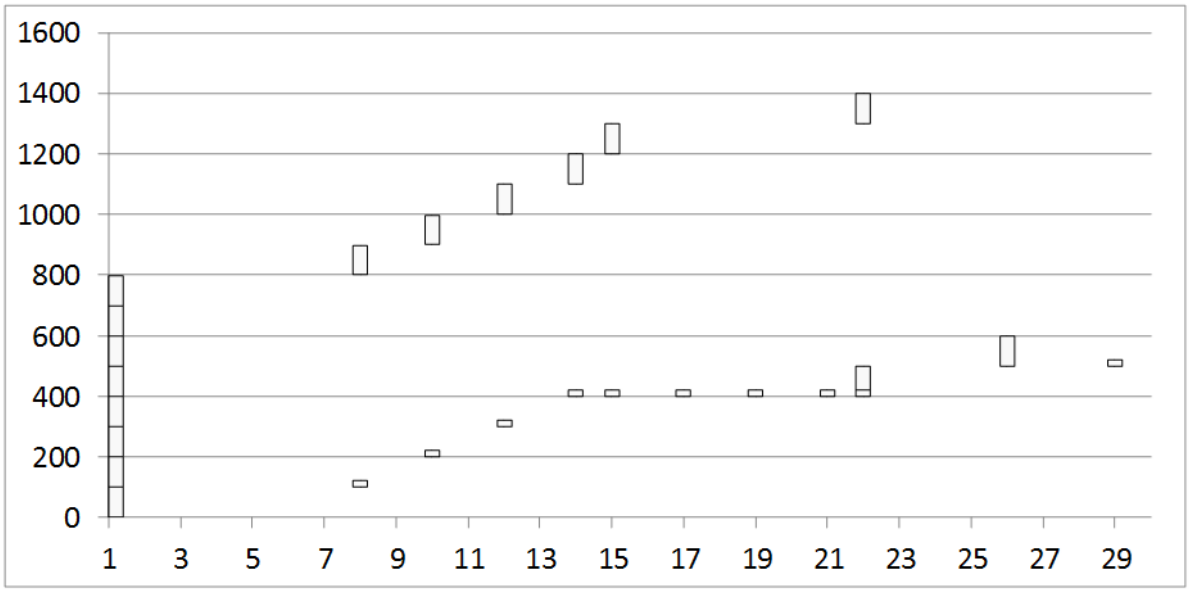
4 (since 3 acks after window size 3 at time 21 have arrived).

3. Now consider a different scenario where the access link is very fast but the next link is slower.




Assume an initial window size of 8 this time. As in the previous question, the window size is reduced to half upon receiving triple duplicate acknowledgements, and it is

incremented by 1 / int(congestion window) upon receiving an acknowledgment. A timeout occurs if the last unacked packet goes unacknowledged for more than 25 time units. The buffer size at the first router is limited, and it follows a drop-tail policy. Assume that all packet losses happen due to buffer overflow at the first router. Answer the following questions:




i. What is the RTT in this case?

From the diagram, the first packet takes a time of roughly 7s/r till the ack is received corresponding to that packet, which can be taken to be the RTT for a single (isolated) packet in the network. To compute it theoretically, transmission across the first link takes s/100r, the second link takes 2s/r, and the third one takes s/r time, for a total of 3.01s/r time. The round trip propagation delay is 4s/r, and since the ack transmission delay is negligible, the total time is 7.01s/r, as needed.

ii. Can you infer the buffer size at the first router? How?

Yes. Note that the dropping happens only when the buffer is full. Since the first link is very fast (on the time scale of s/r), we can assume that the packets are almost instantaneously at the router. Since the first packet dropped is the $5^{th}$ packet, the buffer size is 4.

iii. Why is there no retransmission at time 17 despite a triple dup ack? Why does this retransmission happen later at time 22? Why do two packets get fired off at time 22?

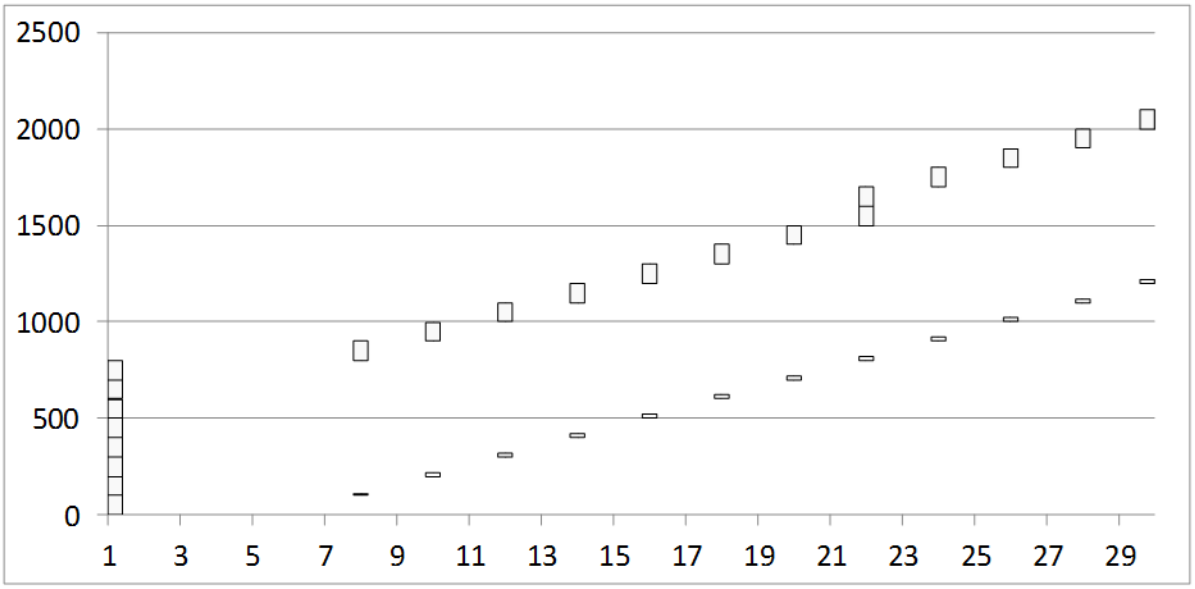
Just before time 17, the window size is 8 + 5/8 = 8.625. Since there is a triple duplicate ack, the window size drops down to 4.3125. Among the 13 packets sent, 6 of them have their acks received just after time 17. So this leaves 7

packets unacknowledged, and thus the sender waits while the inequality (window size + 1) < #(unacknowledged packets) holds. Note that at time 22, the window size goes from 4.8125 to 5.0625 and the number of unacknowledged packets goes down from 4 to 3, so suddenly there are two more vacancies in the set of unacknowledged packets. So it sends two packets (both almost simultaneously because of the fast access link).

iv. When did a timeout occur? Which packet gets timed out?

The timeout occurs at time 26 as can be seen from the retransmission (and noting that the access link is very fast), and the packet 500-599 gets timed out.

4. Consider now a scenario where the buffer size at the first router is very large so that no drops occur. Answer the following questions:




i. What is the round trip time clocked for the first packet? For the fifth packet? For the eighth packet?

From the diagram, we see that the RTT clocked for the 1st, 5th, and 8th packets are 7s/r, 15s/r and 21s/r respectively. From a theoretical perspective, it can be calculated as (RTT for first packet) + (2s/r) * (packet number – 1). This is true because for each extra packet, it stays in the queue for additional 2s/r time as compared to the previous packet (and there is no loss, and the second link is the bottleneck). The RTT for the first packet can be computed from the previous question as well.

ii. When is the window size increased to 9?

At time 22 (one way to verify is to see that there are 2 packets sent over the network instead of just 1). This happens because 8 acks after a window size of 8 increment it by exactly 1.

iii. Since the buffers are assumed to be large enough, there will be no drops and the window size will keep increasing each time a complete round of acknowledgments for the current window are received. What is the problem with such a network?

If there is a long queue, then there will be a larger RTT, so the sender will keep retransmitting data, eventually flooding the network and never sending the complete file either.

iv. Suggest a method to trigger a window size reduction in such a scenario, without witnessing any loss events.

In such a case, there will be a timeout for most packets. Hence we can switch to a scheme that triggers window size reduction on a certain number of timeout events alongside packet losses. This would ensure that whenever the number of timeouts exceeds a certain time, the window size reduces automatically, and thus holding the sender back from flooding the network.