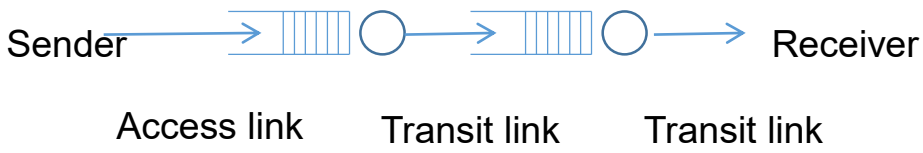
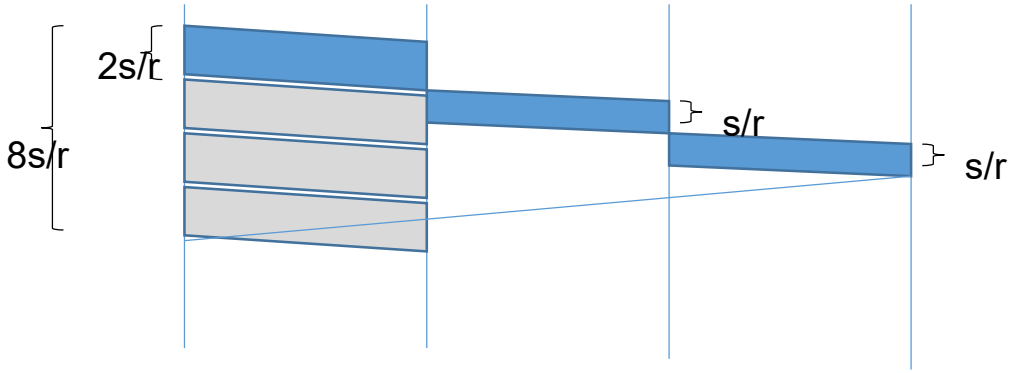
COL-334: Assignment 3, Semester I 2014-2015

Group Members: Yash Gupta, Rishit Sanmukhani, Ujjwal Sinha

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.



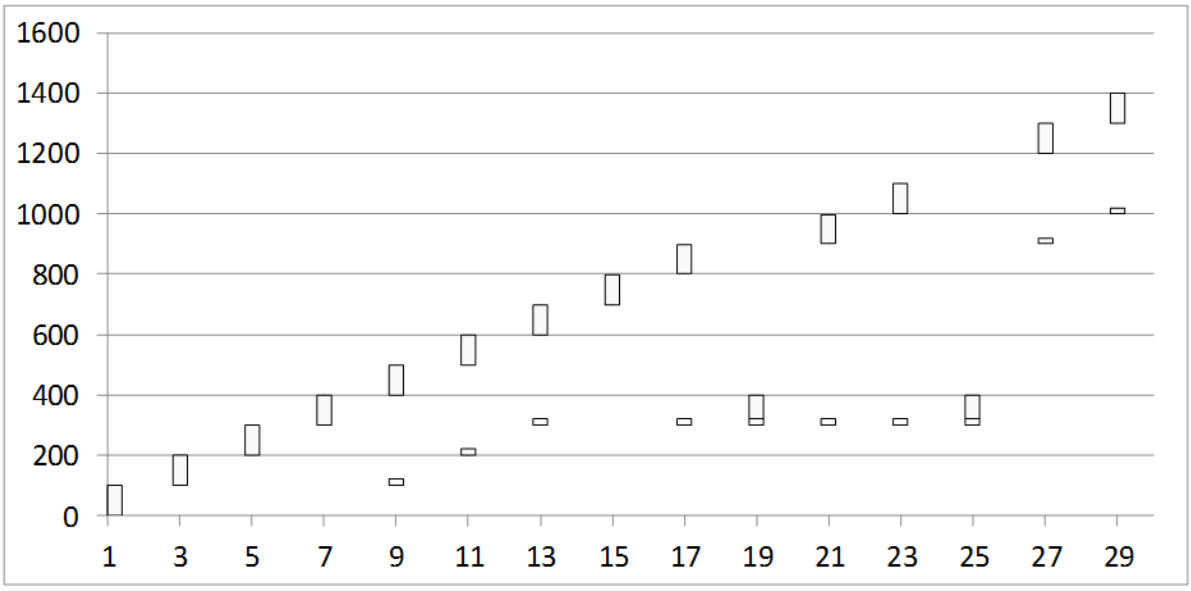


Consider the following packet trace at the sender using a transport protocol similar to TCP. The y-axis indicates sequence numbers in bytes – long vertical rectangles are packets and each packet is 100 bytes long, thus the first packet contains data from sequence number 0 to 99, the second packet from 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.

Assume the congestion window size to be fixed and 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.

Now explain the packet trace below.



i. **Packet 300-399 seems to have been lost. What triggers the retransmission of the lost packet?**
Retransmission has happened because the sender has received 3 duplicate acknowledgements referring to the 4th packet (300-399) at time 13, 17, 19 from the receiver.

ii. **The acknowledgement received at time 21 was generated at the receipt of which packet at the receiver?**
It was generated by the 7th packet (600-699).

iii. **Why is the acknowledgement at time 21 still referring to the lost packet even though it has been retransmitted?**
The lost packet which has been retransmitted at time 19 hasn't reached the receiver before the receiver sends the acknowledgement of the 7th packet referring to the lost packet.

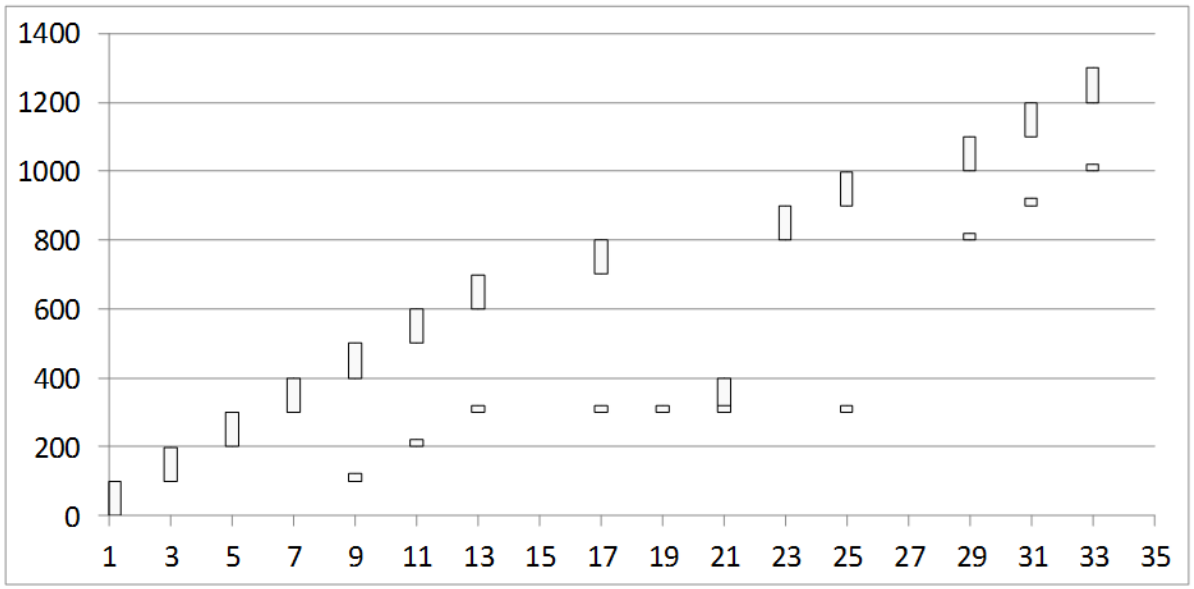
iv. **Why the lost packet is again retransmitted at time 25?**
Upon receiving the three duplicate acknowledgment packet referring to the lost packet at time 21, 23, 25, the sender retransmits the lost packet again at time 25.It seems that the lost packet which was retransmitted at time 19 hasn't reached the receiver yet.

v. **Why is there a sudden jump in the acknowledgement number at time 27? The acknowledgement was generated at the receipt of which packet?**
There is a sudden jump indicating that the receiver has received everything up to 9th packet (800-899) and is expecting the next packet with sequence number starting from 900. This also indicates that the lost packet which was retransmitted at time 19 has reached the receiver.

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: Mentally maintain two variables for congestion window and the outstanding data to understand what is happening.



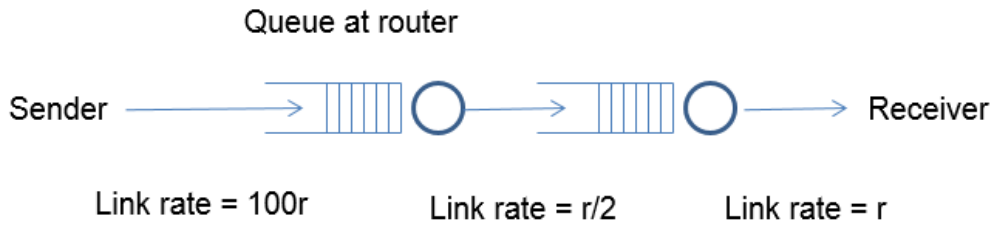
   i. **What is the window size at time 17?**
   Window size at time 17 is 5. (Since 4 acks have been received, w ~= 4 + 4 * ¼)

ii. **What is the window size at time 19? Why is no packet pushed out at time 19? What is the window size at time 21? What is the outstanding data estimated by the sender at time 21?**
Window size at time 19 is 2.5
At time 19, the number of outstanding packet in the network is 2 and since the window size is 2.5, the new packet cannot be fully accommodated in the window, therefore leading to no push out of packets at time 19.
Window size at time 21 is 3. At time 21, the outstanding data is the lost packet which is being retransmitted having the sequence number 300 - 399.

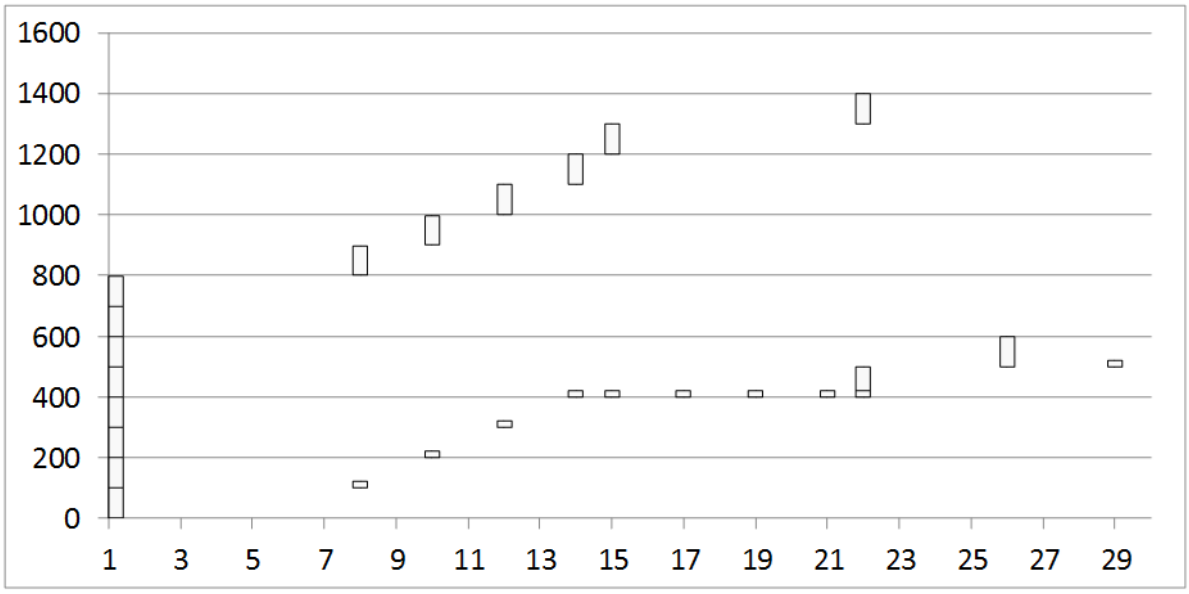
iii. **Why is a packet pushed out at time 23 even though no ack is received at that time?**
At time 23, the window size is 3 and the number of packets in the network is 2. Therefore, one more packet can be pushed into the network as that can be fully accommodated in the congestion window. That's the reason why a packet is pushed out at time 23 even though no ack is received at that time.

iv. **What is the window size at time 31?**
Window size at time 31 is 3.99

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?**
7 units (time between first packet at t=1 and its ack at t=8). Other packets take longer due to slow 2nd link and buffering at fist router.
Also evident from ack received after 7 units (at t=29) for packet 400-499 (sent at t=22) when the network had no pending packets.

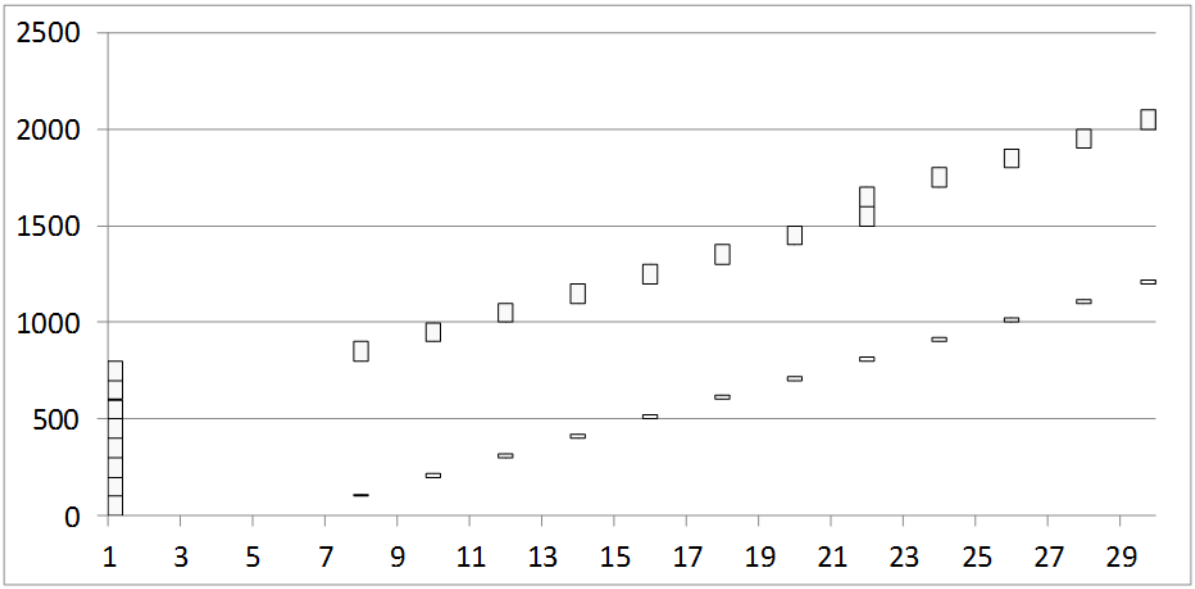
ii. **Can you infer the buffer size at the first router? How?**
Buffer Size is 4. Since the router follows the tail drop policy and 5th packet has been dropped after seeing three duplicate ack and also 6th packet has been dropped after seeing it's timeout at time26, it suggests that the buffer size of the router 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?**
No retransmission happens at time 17, because on receiving 3 dup ack, the window size became 4.312 and at that moment there were 6 packets in the network. For the packet to be transmitted, the number of the packets in the network must be less than the congestion window size, which is not the case at time 17.

At time 22, the window size is 5.065 and the outstanding packet in the network is 3. Therefore the two packets get fired off at time 22.

iv. **When did a timeout occur? Which packet gets timed out?**
The timeout occurs at time 26. Packets 6,7,8 (500 – 799) get 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?**
RTT for 1st packet is 7, 5th packet is 15, 8th packet is 21.

ii. **When is the window size increased to 9?**
Window size is increased to 9 at time 22 (After receiving 8 acks).

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 scenario?**
As the window size keeps on increasing, RTT for packets will also increase due to increasing queue delay at the router. This will lead to timeout after a certain point, and TCP will restart with window size = 1 MSS in slow start mode.
This will lower the throughput achieved as compared to a fast re-transmit scenario with limited buffer size.
Also, due to increasing queue size at buffer, other flows will starve due to timeouts.

iv. **Suggest a method to trigger a window size reduction in such a scenario, without witnessing any loss events.**
We can reduce window size by half whenever the received ack is of a packet transmitted more than 2*RTT time ago. This will allow the buffer to clear up before pushing more packets to the queue.

5. Show that in a steady state TCP connection working in the congestion avoidance phase, the throughput ~ 1.22 x MSS / (RTT x sqrt (L)) where RTT is the round trip time, MSS is the maximum segment size, and L is the loss rate

   Note that in the congestion avoidance phase, all losses are assumed to be detected through fast retransmits and not timeouts, hence the congestion window rises additively and falls to half its value in a saw-tooth pattern.

   ***Hint**: If N packets are sent between two consecutive packet loss events, assume that the events happen due to the loss of only one packet in each event, hence the loss rate can be written as 1/N.*

   Solution:

   At steady state, **the congestion window grows linearly one packet (1 MSS) per RTT** and when there is a loss event, **congestion window is set to half its previous value.**
   Initial congestion window size = c
   Final congestion window size = 2c
   We know that the congestion window grows linearly one packet (1 MSS) per RTT.
   Therefore for congestion window to become 2c from c, it will take c * RTT time.
   Total bytes sent = c * RTT * Throughput. --- (1)
   We know that the number of packets pumped in the network b/w two losses,
   $N = c + c+1 + c+2 + c+3 + \ldots + c+c = c^2 + c(c+1)/2$
   As $c+1 \sim= c$, $N = 1.5*c^2$

   Total bytes sent = $N * MSS = (1.5*c^2) * MSS$ --- (2)

   Equating (1) and 2 and substituting c in terms of L (loss rate)
   $L = 1/N = 1 / (1.5*c^2)$
   On rearranging, $C = 0.812 / sqrt(L)$ --- (3)

   $c * RTT * Throughput = (1.5*c^2) * MSS$
   $Throughput = 1.5*c*MSS/ RTT = 1.5*0.812*MSS/ (L)*RTT)$
   $= 1.22*MSS/(RTT*sqrt(L))$

6. Two TCP flows are running through the same bottleneck router through a shared buffer, and competing with each other. When the buffer gets congested, it drops a packet each from both the flows, and both the flows halve their windows simultaneously. The RTT of the first flow is half the RTT of the second flow. Show that the first flow settles at twice the bandwidth of the second flow at steady state.

***Hint**: Start with writing recursive equations for the window sizes of each flow in terms of the i^{th} congestion event.*
$W^1_{i+1} = (W^1_i + T)/2$
$W^2_{i+1} = (W^2_i + T/2)/2$
Where T is the number of transmission rounds between two consecutive loss events in which a window size amount of data is dispatched.

Solution:

Let W_i and W'_i be the congestion window sizes after i^{th} congestion event and W and W' be congestion window sizes after long time i.e. $W = W_i$ when $W_{i+1} = W_i$ similarly W'
T_i be the number of transmission rounds between i^{th} and $i+1^{th}$ congestion event.

$W_n = T_n/2 + T_{n-1}/4 + T_{n-2}/16 + \ldots T_1/2^n + W_1/2^n$
$W'_n = T_n/4 + T_{n-1}/8 + T_{n-2}/32 + \ldots T_1/2^n + W'_1/2^n$

For large n, $W_n = 2 \cdot W'_n$, hence the bandwidth of second is half of the first connection.