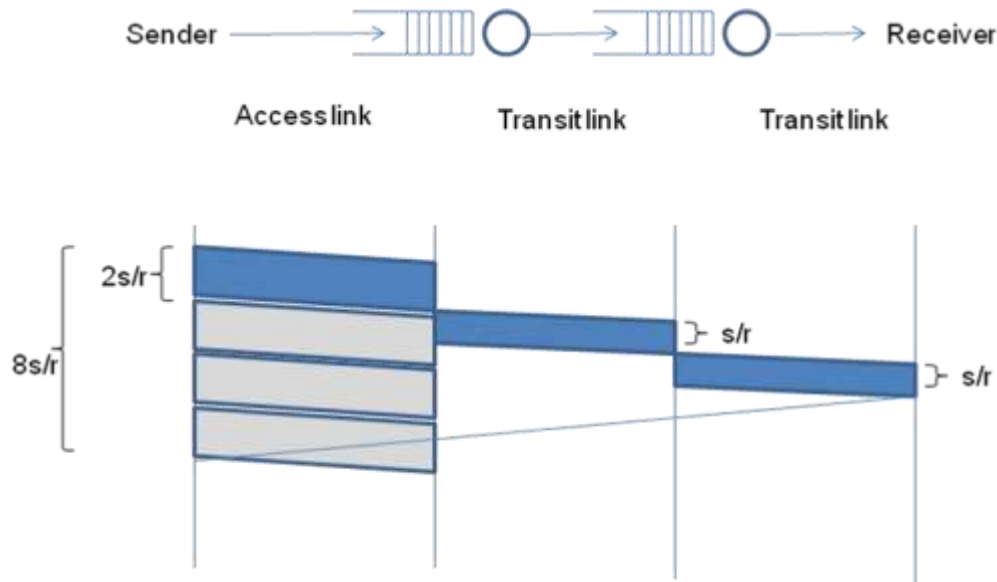


COL-334/CSL-374/CSL-672: Assignment 3, Semester 2014-2015 I

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.

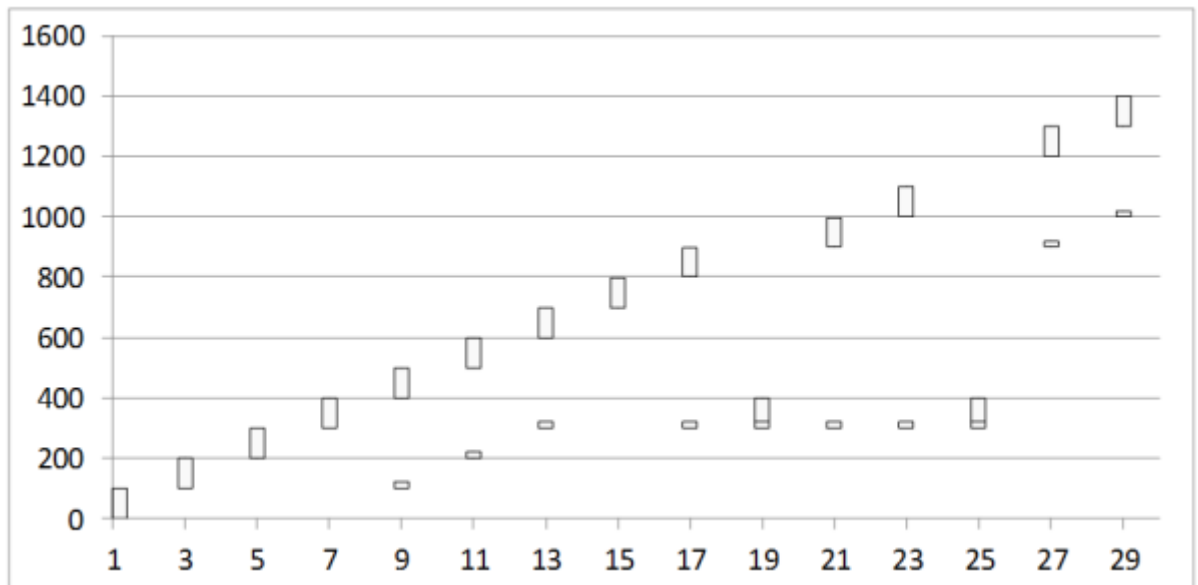


Consider the following packet trace at the sender using a transport protocol similar to TCP. The y-axis indicates sequence numbers in bytes – long vertical rectangles are packets and each packet is 100 bytes long, thus the first packet contains data from sequence number 0 to 99, the second packet from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

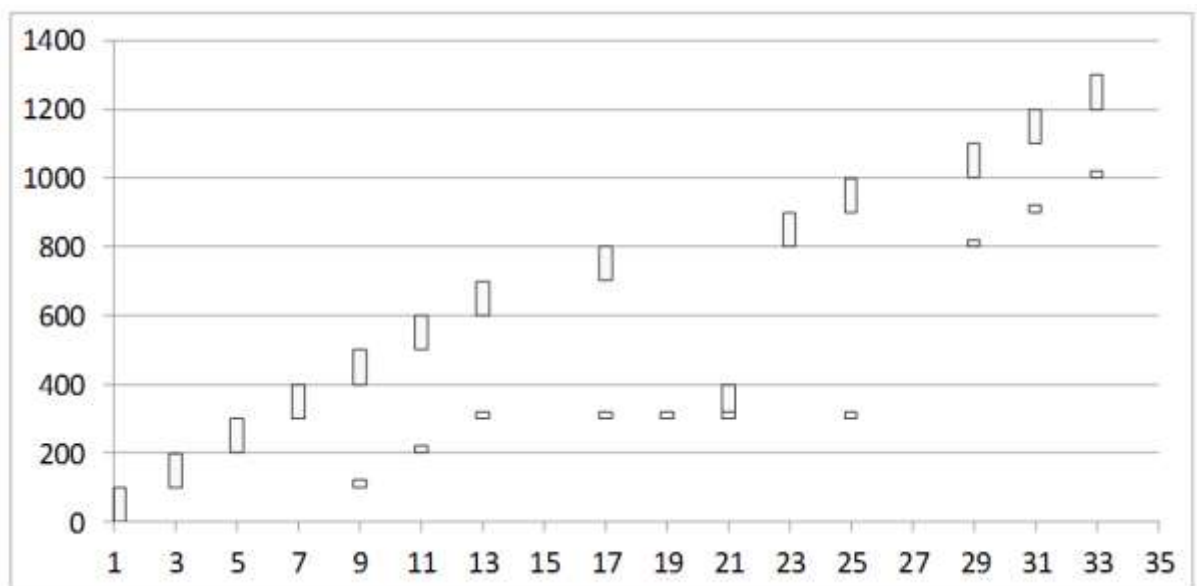


- i. Packet 300-399 seems to have been lost. What triggers the retransmission of the lost packet?
- ii. The acknowledgement received at time 21 was generated at the receipt of which packet at the receiver?
- iii. Why is the acknowledgement at time 21 still referring to the lost packet even though it has been retransmitted?
- iv. Why is the lost packet again retransmitted at time 25?
- v. Why is there a sudden jump in the acknowledgement number at time 27? The acknowledgement was generated at the receipt of which packet?

2. In another variant, the initial congestion window size is started at 4 packets, and the window is incremented fractionally by $(1 / \text{int}(\text{current window size}))$ 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.

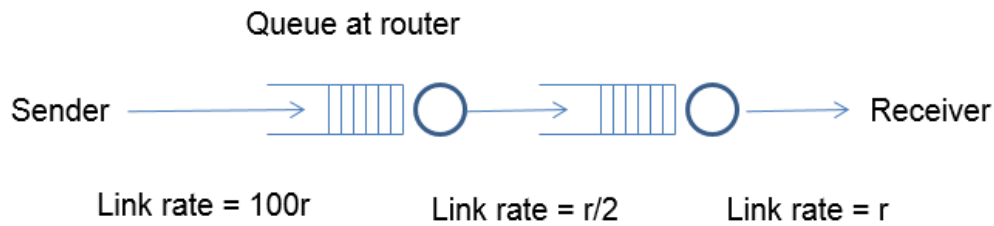
The window size also reduces to half upon receiving triple duplicate acknowledgements which is seen as an evidence of packet loss. Note that receipt of the third dup ack will not increment the window, ie. if the window is 5 when the third dup ack arrives, it will just be reduced to 2.5, and not add another $1 / \text{int}(2.5)$ 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.

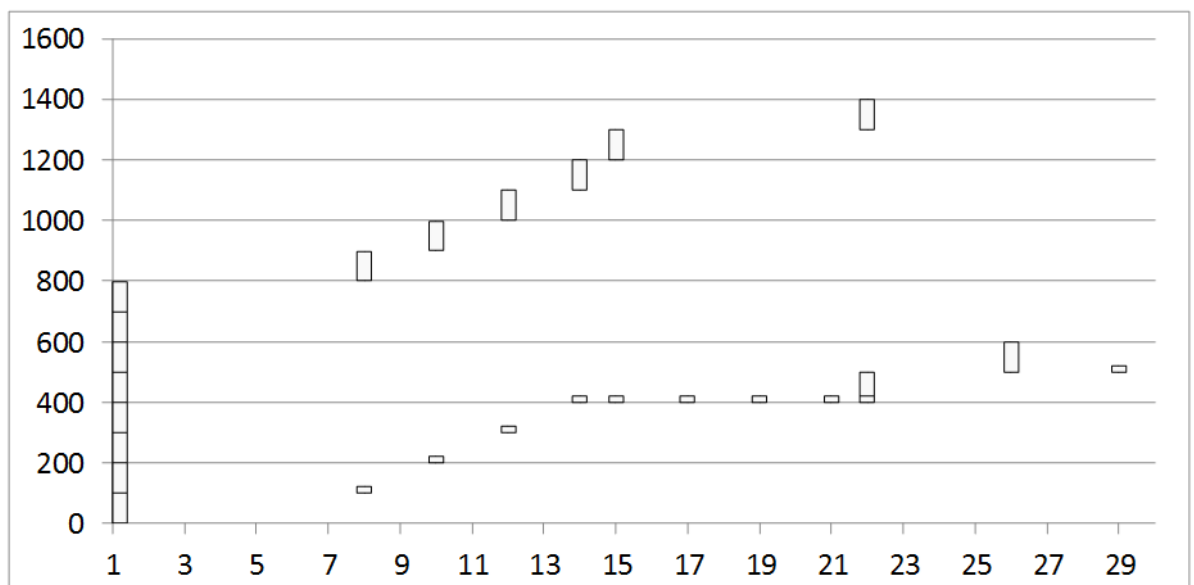


- i. What is the window size at time 17?
- ii. What is the window size at time 19? Why is no packet pushed out at time 19? What is the window size at time 21? What is the outstanding data estimated by the sender at time 21?
- iii. Why is a packet pushed out at time 23 even though no ack is received at that time?
- iv. What is the window size at time 31?

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

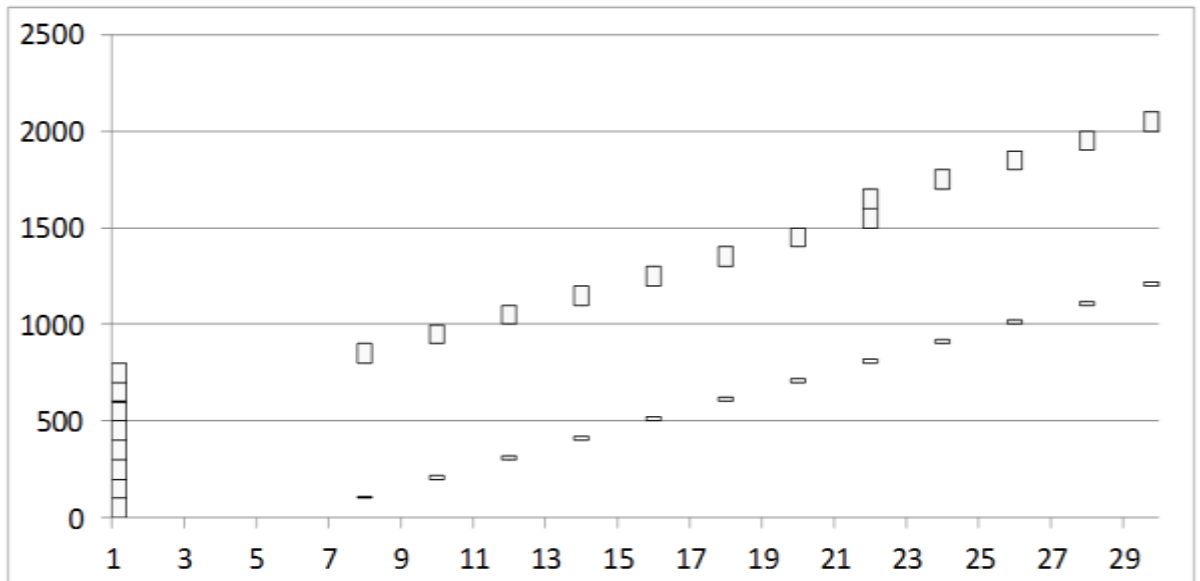


Assume an initial window size of 8 this time. As in the previous question, the window size is reduced to half upon receiving triple duplicate acknowledgements, and it is incremented by $1 / \text{int}(\text{congestion window})$ upon receiving an acknowledgment. A timeout occurs if the last unacked packet 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:



- What is the RTT in this case?
- Can you infer the buffer size at the first router? How?
- 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?
- When did a timeout occur? Which packet 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:



- What is the round trip time clocked for the first packet? For the fifth packet? For the eighth packet?
- When is the window size increased to 9?
- 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?
- Suggest a method to trigger a window size reduction in such a scenario, without witnessing any loss events.

5. Show that in a steady state TCP connection working in the congestion avoidance phase, the throughput $\sim \frac{1.22 \times \text{MSS}}{\text{RTT} \times \sqrt{L}}$ 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 $1/N$.

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.

$$W_{i+1}^1 = (W_i^1 + T) / 2$$

$$W_{i+1}^2 = (W_i^2 + T/2) / 2$$

Where T is the number of transmission rounds between two consecutive loss events in which a window size amount of data is dispatched.