CSL-374/672: Computer Networks, Semester 2014-2015 I
Minor-2 take home component

Name: SHIVAM GARG

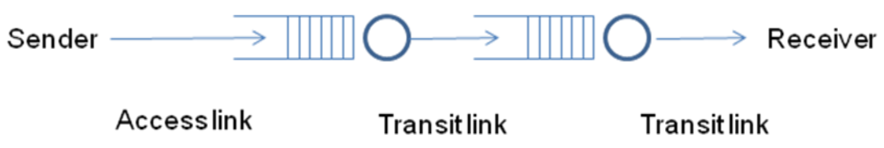
Entry #: 2012CS10252

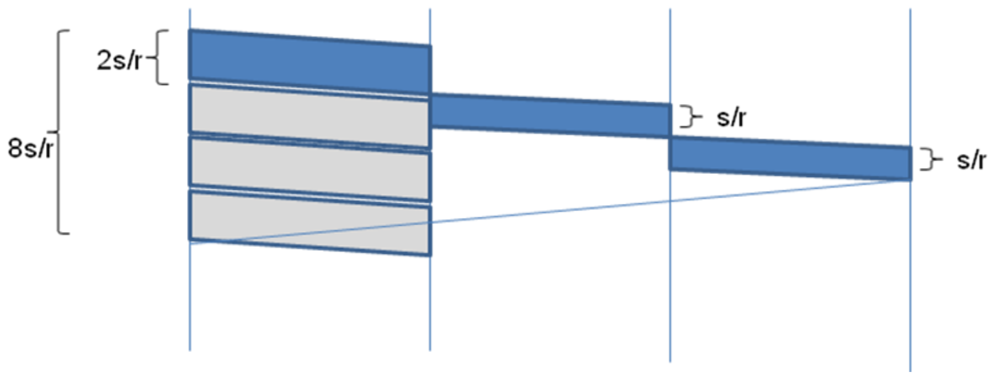
Evaluation (leave blank)

1		6	
2		7	
3		8	
4		9	
5		10	

Total (out of 43): ______________________________________

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.



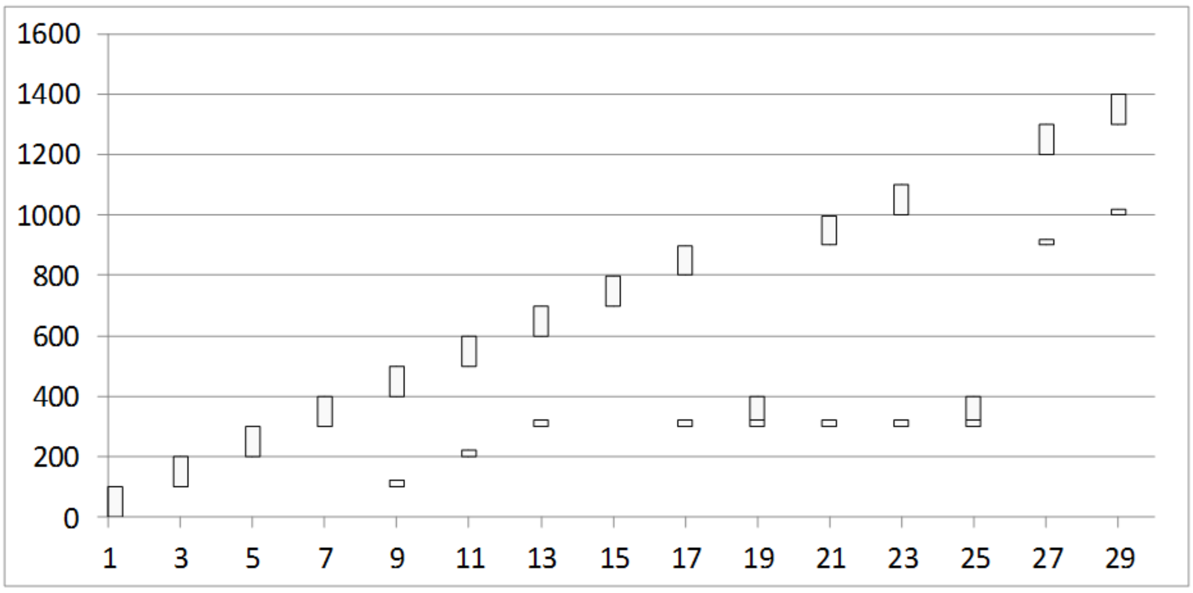


Consider the following packet trace at the sender using a transport protocol similar to TCP. The y-axis indicates sequence numbers in bytes – long vertical rectangles are packets and each packet is 100 bytes long, thus the first packet contains data from sequence number 0 to 99, the second packet from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



i. Packet 300-399 seems to have been lost. What triggers the retransmission of the lost packet? [1]

ANSWER:

The duplicate ack received at time 19 for byte range 200-299 is the third duplicate ack. So by fast retransmit policy , TCP resends the Packet 300-399.

ii. The acknowledgement received at time 21 was generated at the receipt of which packet at the receiver? [1]

ANSWER:

It was generated at the receipt of Packet 600-699 since at time 21 , six acks have been received and there has been no re-ordering and further the packet 300-399 has been lost, the sixth packet is 600-699.

iii. Why is the acknowledgement at time 21 still referring to the lost packet even though it has been retransmitted? [1]

ANSWER:

Since the time taken to reach receiver by a packet is 4 units and retransmission has been done at 19th unit of time. The packet is yet to be received by the receiver.

iv. Why is the lost packet again retransmitted at time 25? [1]

ANSWER:

Since the tri duplicate ACKS at time 21,23,25 are received , the sender resends the packet 300- 399 as it has been still not been acked. These ACKS were released for the packets transmitted before retransmission of Packet 300-399.

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]

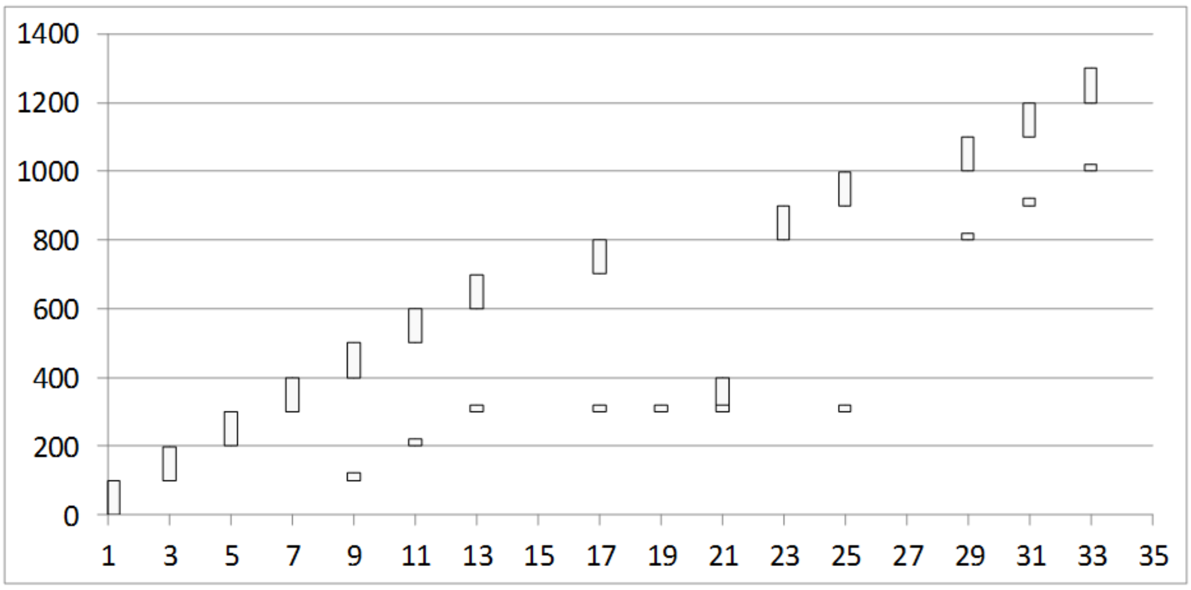
ANSWER:

The sudden jump at time 27 is due to the receipt of Packet 300-399 at time 23 by receiver and consequently, the sender transmits cumulative ack for all packets that have been received till that point of time.

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? [1]

ANSWER:

The window size will be 4.25(4+1/4) at time 9, 4.5(4.25+1/4) at time 11, 4.75(4.5+1/4) at time 13, 5(4.75+1/4) at time 17 due to successive acks being received. The window size is **5** at time 17.

ii. What is the window size at time 19? Why is no packet pushed out at time 19? [2]

ANSWER:

The size of window is reduced to 2.5 since third duplicate arrives at time 19 and window size at time 17 is 5 . The no. of outstanding packets is 2 as four packets were outstanding at time 17 and at time 19 triple duplicate acks were received which implied 1 packet was dropped and 1 ack was received which reduced outstanding packets by one, therefore outstanding packets are 2. The window size is 2.5, therefore the window is already full and no packet can be sent.

iii. What is the window size at time 21? What is the outstanding data estimated by the sender at time 21? [2]

ANSWER:

Size of window at time 21 is 3 as at time 19 the size is 2.5 and with ack being received at time 19 window size was increased to 3 by adding 0.5 to 2.5. The outstanding data estimated by sender is 2 packets i.e. 200 bytes.

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

ANSWER:

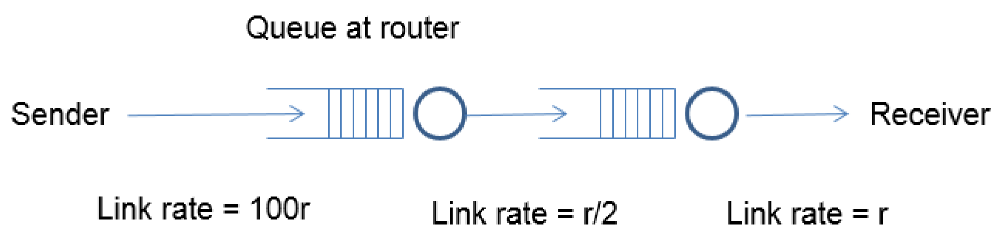
Since the window size is 3, a more packet can be accommodated to be sent as outstanding packets is 2 at that point.

v. What is the window size at time 31? [1]

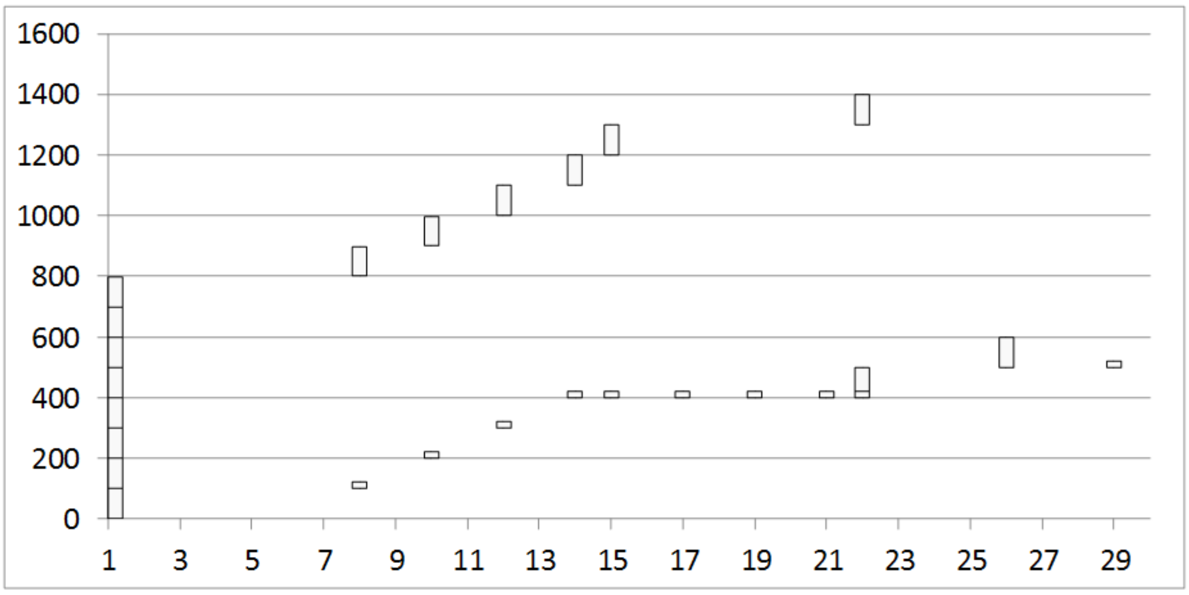
ANSWER:

The window size at 23 is 3 and due to acks being received at time 25,29 and 31 window size is increased by 1/3 3 times, therefore window size becomes 4.

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? [1]

ANSWER:

RTT=(s/100r+2s/r+s/r)/(s/r)+propagation delay

= 3.01 + 4 units of time

7.01 time units.

ii. Can you infer the buffer size at the first router? How? [2]

ANSWER:

The buffer size is 400 bytes(4 packets) as all losses happen due to buffer of first router and packet with 400-499 range was dropped and was the first packet to be dropped therefore due to tail drop policy of buffer , the buffer was full and could not accommodate packet 400-500.

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? [2]

ANSWER:

The window size was reduced by half to 4.3125 and the no. of outstanding packets in the network at that point is 6 which is greater than size of window. Therefore, no retransmission will take place.

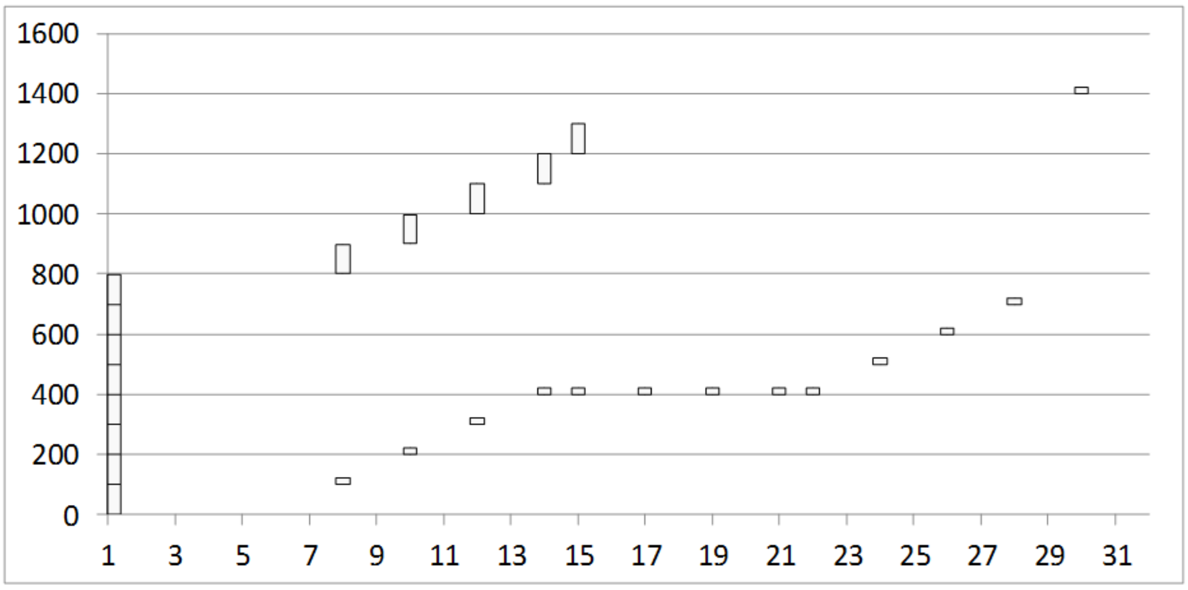
The window size is 5+(1/16) at time 22 and the number of outstanding packets is 3 therefore two packets can be sent out of which one was to be retransmitted.

iv. When did a timeout occur? Which packet gets timed out? [2]

ANSWER:

Timeout occurs at time unit 26. Packet 500-600 got timed out as it was sent at time unit 1 and it was not acked until time unit 26.

v. Timeouts are bad because the congestion avoidance algorithms push down the window to very low values, and it also takes much longer to detect a loss through a timeout. If selective acknowledgements were being used, then the duplicate acknowledgements would have also contained information about the packet they were actually acknowledging, and therefore the triple duplicate acknowledgement at time 17 could have helped indicate that packets 400-499/500-599/600-699/700-799 were all lost. This would have consequently implied that the number of outstanding packets is lesser by 4, and would have triggered retransmissions sooner and not led to a timeout. Complete the plot below in this case by drawing the data packets, assuming that the acknowledgements carry selective acknowledgement numbers of which data packets they are acknowledging, and that triple duplicate acknowledgements imply a loss of all unacknowledged data prior to the selective acknowledgement numbers. [3]



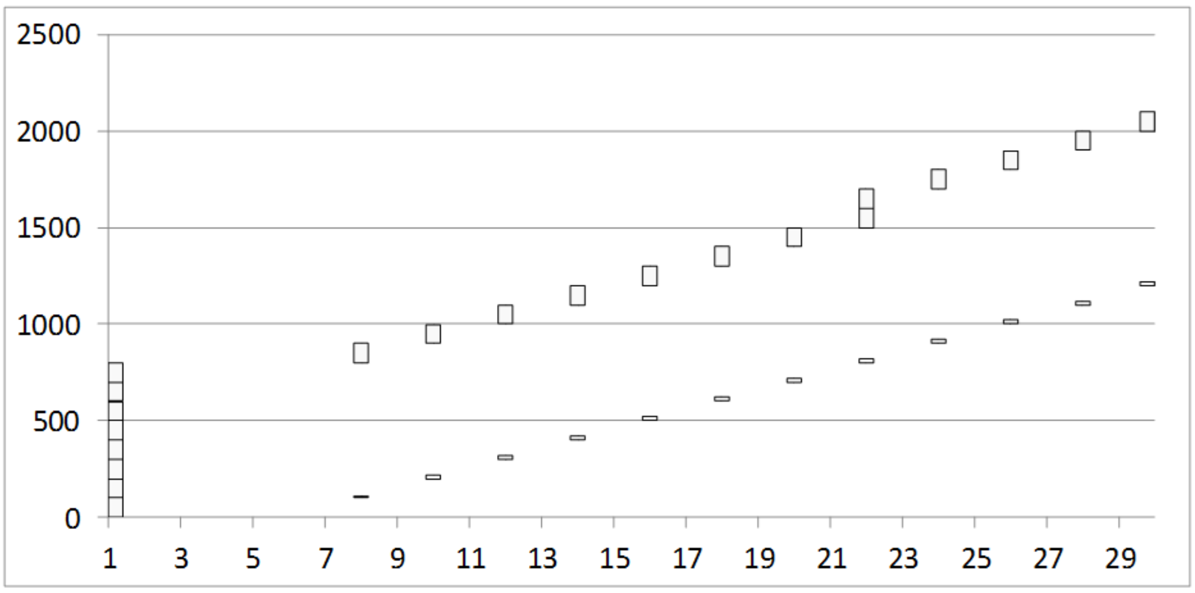
ANSWER:

Time	Packets sent	Congestion Window Size	Outstanding Packets	Acks received for packets
1	0-99,100-199,200-299,300-399,400-499,500-599,600-699,700-799	8	8	-
2	-	8	8	-
3	-	8	8	-
4	-	8	8	-
5	-	8	8	-
6	-	8	8	-
7	-	8	8	-
8	800-899	8+(1/8)	8	0-99 cumulative for 0-99
9	-	8+(1/8)	8	-
10	900-999	8+(2/8)	8	100-199 cumulative for 100-199
11	-	8+(2/8)	8	-
12	1000-1099	8+(3/8)	8	200-299 cumulative for 200-299
13	-	8+(3/8)	8	-
14	1100-1199	8+(4/8)	8	300-399 cumulative for

				300-399
15	1200-1299	8+(5/8)	8	800-899 cumulative for 300-399
16	-	8+(5/8)	8	-
17	400-499	4+(5/16)	4	900-999 cumulative for 300-399
18	-	4+(5/16)	4	-
19	500-599	4+(5/16)+(1/4)	4	1000-1099 cumulative for 300-399
20	-	4+(5/16)+(1/4)	4	-
21	600-699	4+(5/16)+(2/4)	4	1100-1199 cumulative for 300-399
22	700-799,1300-1399	5+(1/16)	5	1200-1299 cumulative for 300-399
23	-	5+(1/16)	5	-
24	1400-1499	5+(1/16)+(1/5)	5	400-499 cumulative for 400-499
25	-	5+(1/16)+(1/5)	5	-
26	1500-1599	5+(2/5)+(1/16)	5	500-599 cumulative for 500-599
27	-	5+(2/5)+(1/16)	5	-
28	1600-1699	5+(3/5)+(1/16)	5	600-699 cumulative for 600-699
29	-	5+(3/5)+(1/16)	5	-
30	1700-1799	5+(4/5)+(1/16)	5	700-799 cumulative for 1300-1399

The window size increases linearly till time 17 since all no triple duplicate acks have been received. At time 17, sender comes to know that all packets from 400-499 till 799-800 were lost as tri duplicate acks were received for packet cumulative acks 300-399. Thus the sender queues the all four lost packets for transmission. Due to the loss of four packets at time 17, the sender makes the size of window to half.
Since no loss has been observed for the all possible time intervals the window size keeps on increasing and at time 30 1400 bytes of data will be received.

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? [1]

ANSWER:

The round trip time for the first packet is 7 (8-1) units of time as the ack of first packet is received at 8. Fifth packet is 15 (16-1) units of time and for eighth is 21 (22-1) units of time.

ii. When is the window size increased to 9? [1]

ANSWER:

When the eighth ack is received i.e. at 22 units of time as after receiving 8 acks the window size has increased by 1/8 8 times .

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? [2]

ANSWER:

The queueing delay increases in case of infinite buffer capacity. As more packets are dispatched the RTT of packets increase due to queueing delay as the queue size increases which can cause premature timeouts and consequently addition of more packets by the sender thus increasing the traffic in network and triggering even more queueing delays. This will lead to underutilisation of network.

iv. Suggest a method to trigger a window size reduction in such a scenario, without witnessing any loss events. [2]

ANSWER:

The RTT increases with increase in window size. When RTT and transmission delay of packets increase, the packets are more likely to get timed out. Whenever RTT and transmission delay is more than timeout value, the window should be reduced. Thus constant monitoring of RTT is required and when RTT and transmission delay are increased, the window should be reduced to half and increased linearly similar to TCP window size so that timeouts don't happen and the window size will be stabilised to the optimum value when RTT stabilises.

We hope this would help you understand some interesting interplays noticed in transport layer protocols with larger router buffers and heterogeneous link rates!

5. Long distance wireless links are susceptible to bit errors that can lead to packet corruption.

a. The bit error rate on a wireless link = p_B, ie. the probability that a bit under transmission may get corrupted. Show that the packet error rate p_P for large packets of s bits = $s.p_B$, if the bits errors are independent of each other. [2]

ANSWER:

Probability that a bits is not corrupted=$1-p_b$

Since probability that bits are corrupted is independent of each other, therefore probability that packet is corrupted= 1- Prob(each bit is not corrupted)

Prob(each bit is not corrupted)=$(1-p_b)^s$

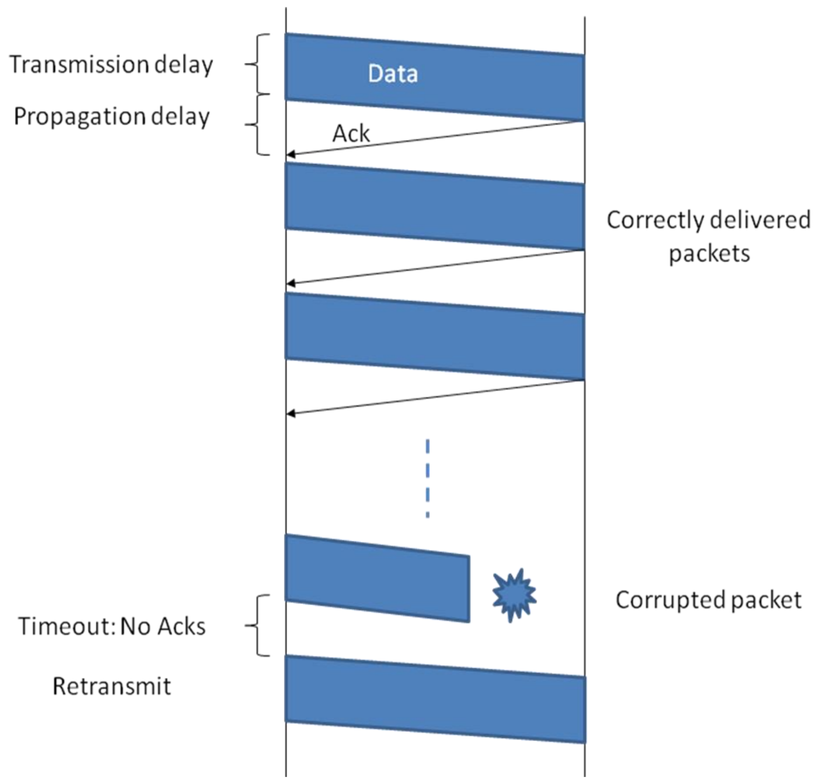
Prob(packet is corrupted)=$1-(1-p_b)^s$

Since $p_b < 1$ therefore $(1-p_b)^s \approx 1-sp_b$

Therefore Prob(packet corrupted)=$1-(1-sp_b)$

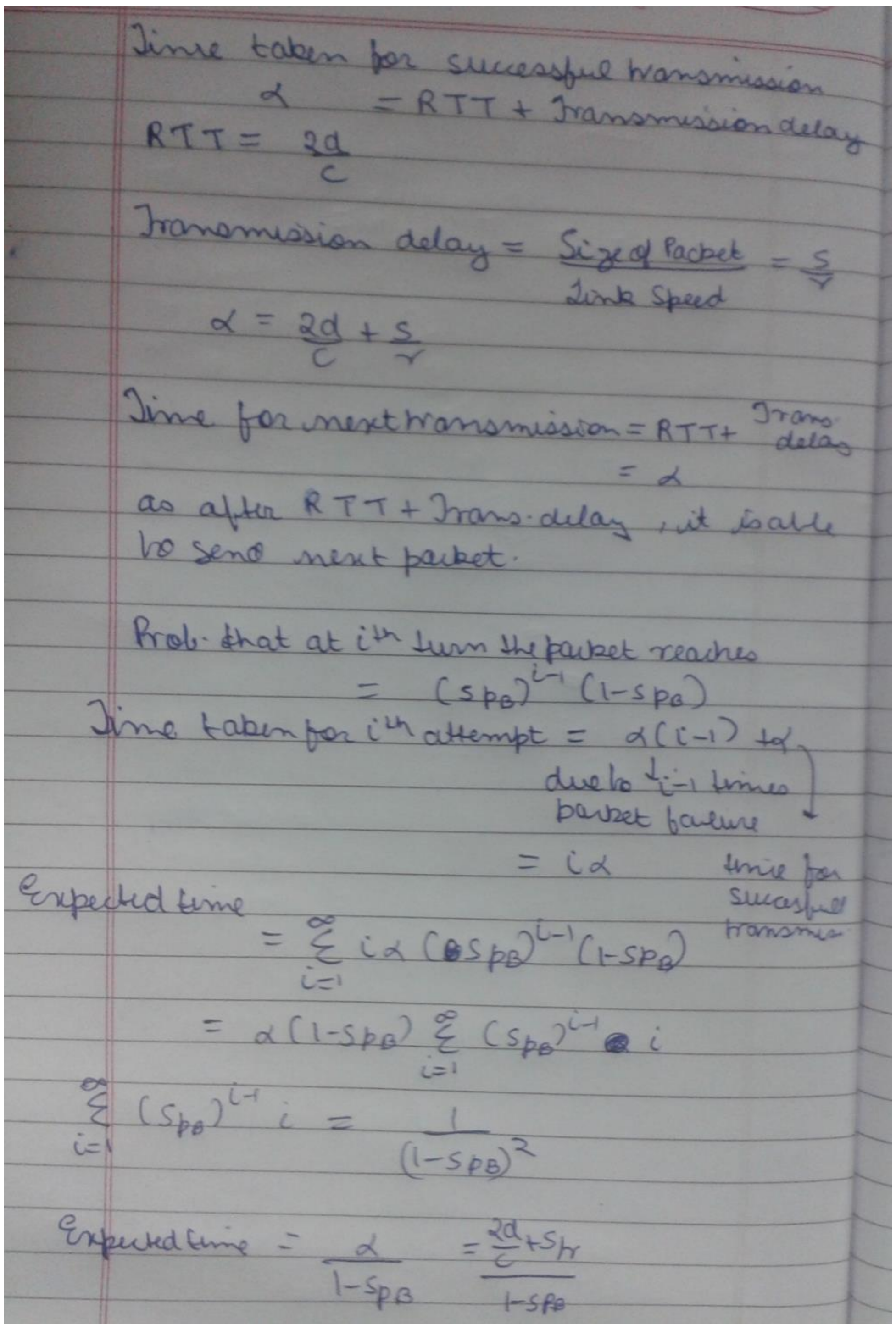
Prob(packet corrupted) = sp_b

b. A stop-and-wait acknowledgement protocol is used at the link layer to improve reliability. The protocol works as follows: an acknowledgement is sent for every packet; if the Ack is not received within a maximum timeout then the packet is retransmitted.



What is the network utilization (ie. throughput) of this scheme, in terms of the bit error rate p_B, data packet size s, transmission rate r, one way propagation distance = d, and speed of propagation = c. Assume that the timeout = maximum propagation round trip delay. And that the acks are too small in size and do not get corrupted. [3]

ANSWER:



Time taken for successful transmission

α = RTT + Transmission delay

$RTT = \frac{2d}{c}$

Transmission delay = $\frac{\text{Size of Packet}}{\text{Link Speed}} = \frac{s}{r}$

$\alpha = \frac{2d}{c} + \frac{s}{r}$

Time for next transmission = RTT + Trans. delay

$= \alpha$

as after RTT + Trans. delay, it is able to send next packet.

Prob. that at ith turn the packet reaches

$= (s p_B)^{i-1}(1 - s p_B)$

Time taken for ith attempt $= \alpha(i-1) + \alpha$

due to i-1 times packet failure

time for successful transmis

$= i\alpha$

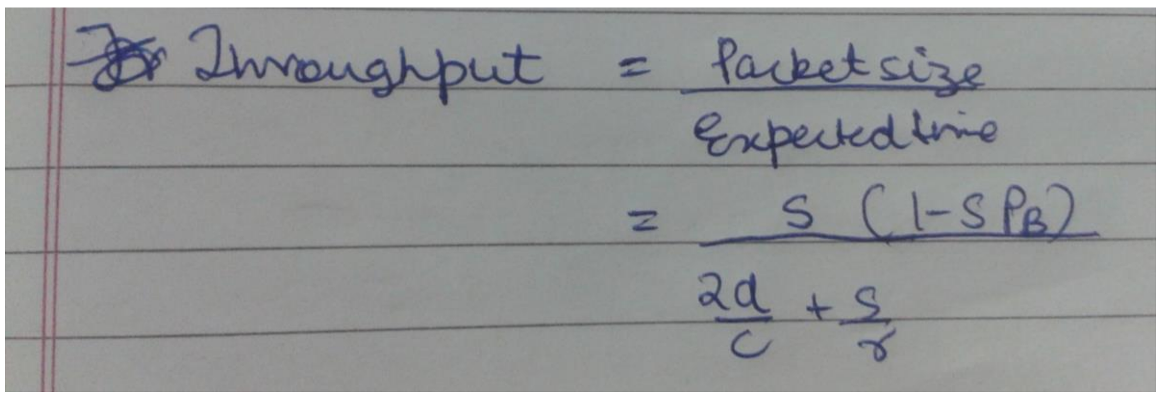
Expected time

$$= \sum_{i=1}^{\infty} i\alpha (s p_B)^{i-1}(1 - s p_B)$$

$$= \alpha(1 - s p_B)\sum_{i=1}^{\infty} (s p_B)^{i-1} i$$

$$\sum_{i=1}^{\infty} (s p_B)^{i-1} i = \frac{1}{(1 - s p_B)^2}$$

Expected time $= \frac{\alpha}{1 - s p_B} = \frac{\frac{2d}{c} + \frac{s}{r}}{1 - s p_B}$

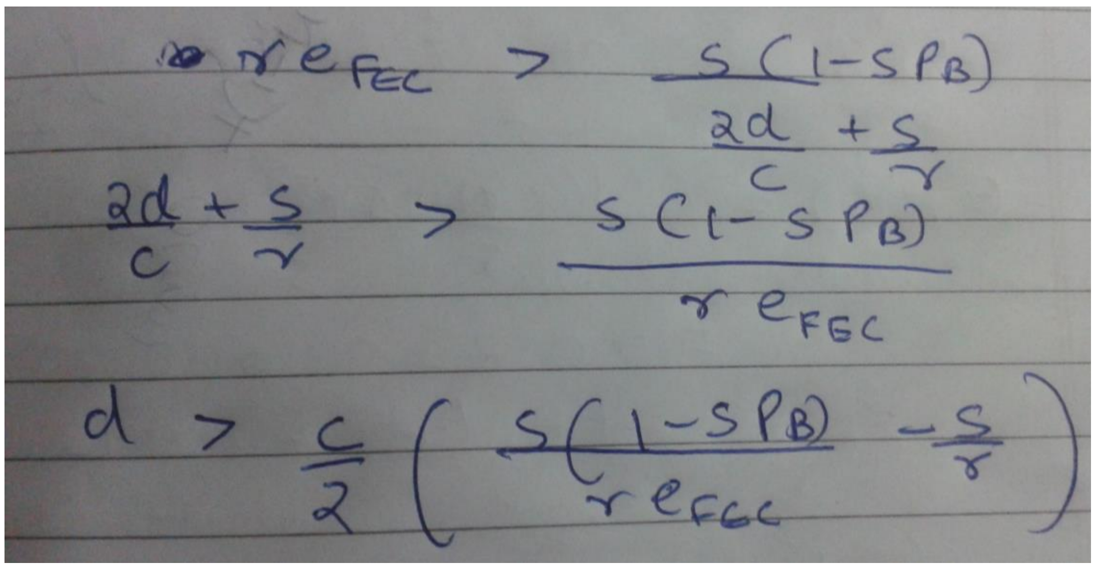


Throughput $= \frac{\text{Packet size}}{\text{Expected time}}$

$$= \frac{s(1 - s p_B)}{\frac{2d}{c} + \frac{s}{r}}$$

c. An alternate scheme called FEC (Forward Error Correction) encodes the packets so that for every m packets, (m + k) encoded packets are sent such that if any m of these (m + k) packets are received correctly then the original m packets can be reconstructed. No acknowledgements are used – the sender simply blasts away the encoded packets at its transmission rate. The network utilization of this scheme is clearly m r / (m + k). Use the previous result to express the propagation distance d in terms of e_{FEC} = m / (m + k) for which the throughput with using FEC will be better than the throughput of the previous acknowledgement scheme. [1]

ASNWER:



$$r e_{FEC} > \frac{s(1-sP_B)}{\frac{2d}{c}+\frac{s}{r}}$$

$$\frac{2d}{c}+\frac{s}{r} > \frac{s(1-sP_B)}{r e_{FEC}}$$

$$d > \frac{c}{2}\left(\frac{s(1-sP_B)}{r e_{FEC}} - \frac{s}{r}\right)$$

re_{FEC} is the throughput of Forward Error Correction. d is the propagation distance. Units of d are in m if c is in ms^{-1} , s is in bytes and r is in bytes per sec.

d. Evaluate the expression above for e_{FEC} = 2/3, bit error rate p_B = 0.00001, propagation speed c = 3 x 10^8 m/s, packet size s = 1000 bytes, and transmission rate r = 5 Mbps. What does this tell you about reliable transmission over long distance wireless links? [2]

ANSWER:

Putting the values in the above formula

d>91200 m i.e. 91.2 km

Communication over long links takes a lot of time in stop and wait protocol as the propagation time increases linearly with propagation distance and throughput is inversely proportional to propagation distance. So a large time is wasted in sending packet to receiver as propagation delay even when bandwidth available is large .In FEC the throughput does not depend on propagation distance and becomes favourable at higher distances.

e. Alternately, for such scenarios where links are long and propagation delays are high because of which stop-and-wait protocols are inefficient, can you propose a pipelined method with acknowledgements where a large number of packets can be maintained in flight? How would you dynamically adjust the number of packets in flight if you did not know the actual round trip propagation delays in advance?[2]

ANSWER:

In a pipelined version, the no. of packets which can be sent in a batch is more than one. This approach will help increase the efficiency of the network by a factor approximately equal to batch size.

Efficiency = ns/(ns/r+2d/c)

=1-(2d/c)(ns/r+2d/c)

Thus the efficiency is directly related to batch size. But this result is valid as it assumes there are no losses in the network. If losses happen, then the efficiency decreases and probability of packet getting lost is directly proportional to batch size. So something similar to TCP will work here.

A window size will be maintained which will signify the batch size. The acknowledgements will be sent at the receipt of last packet which will be a cumulative acknowledgement. Retransmission will depend on the buffers at the receiver according to the no. of packets which the buffer at receiver can store. Each duplicate ack will convey the sequence number of packet for which the ack was generated and the sequence number upto which all packets have been received. Sender can determine the packets which have been lost specifically and retransmit the same.

The window size will be determined dynamically as in TCP by following additive-increase-multiplicative-decrease protocol. AIMD will take care of RTT's and packet losses. If the RTT becomes too high, then timeout will happen and the window size will reduce to 1 and if duplicate acks are received then window size is reduced to half.