Computer Communications and Networks (COMN) Course, Spring 2014

Coursework: Results Sheet

Forename and Surname:	Ignotas Sulzenko
Matriculation Number:	S1137931

Question 1 – Impact of retransmission timeout on number of retransmissions with stop-and-wait protocol.

Retransmission timeout (ms)	Number of re-transmissions	Throughput (Kilobytes per second)
10	6604	23.5016
20	2587	22.3384
30	2494	21.1989
40	504	20.0929
50	440	19.6850
60	498	18.3030
70	485	17.6454
80	439	17.4713
90	473	16.4266
100	456	16.0320

Question 2 – How does the throughput behave when the retransmission timeout increases and why? What is the optimal value for retransmission timeout?

Both the number of packet re-transmissions and the throughput decreases as the retransmission timeout increases. The number of re-transmissions decreases because although some of the packets are claimed as "lost", they eventually arrive and the sender does not have to re-transmit them anymore. The throughput decreases because for the lost packets we wait longer until we "realize" that the packet is indeed lost and needs to be resent, which means that we will spend a longer time sending the file.

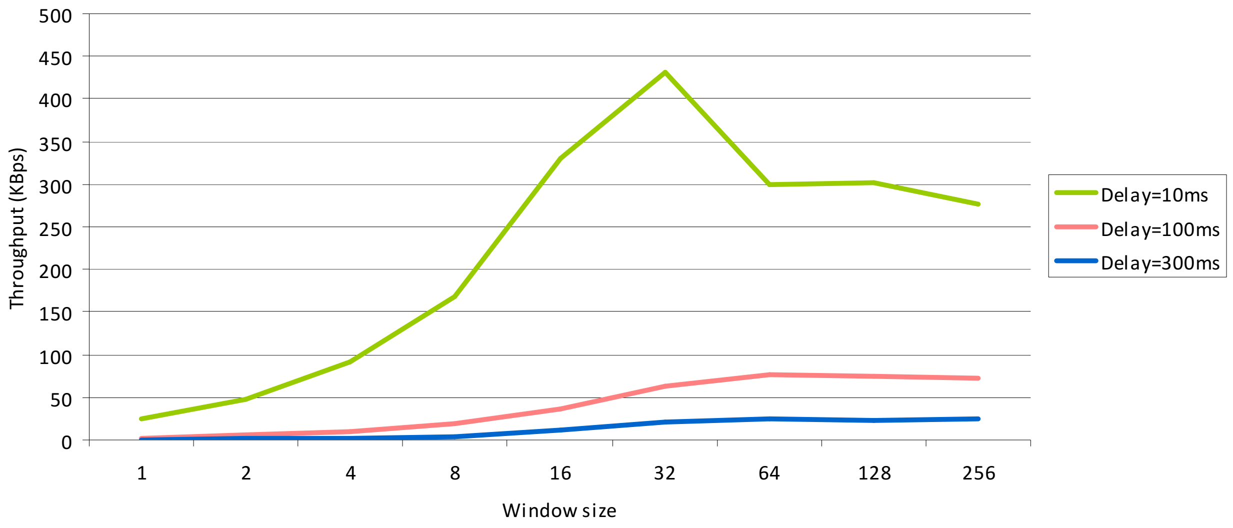
I chose the re-transmission rate of 40ms, because after the retransmission rate of 40ms, the number of re-transmissions made stops to significantly decrease. This allows us to minimize the bandwidth used for retransmissions and only send the retransmissions for the packets that are actually lost, while at the same time, maximizing the highest possible throughput.

I have done similar experiments to find the optimal value for retransmission timeout for the network delays of 100ms and 300ms. In both cases I found out that the optimal retransmission timeout is the round-trip-time + small constant to deal with the time required for the data processing in the receiver.

Question 3 – Experiments with Go-Back-N:

	Throughput (Kilobytes per second)		
Window Size	**Delay = 10ms**	**Delay = 100ms**	**Delay = 300ms**
Optimal Retransmission timeout(ms)	40	220	620
1	24.3702	2.4905	0.8421
2	48.3251	4.9758	1.6642
4	91.9942	9.2941	2.13912
8	168.2534	19.2341	4.1293
16	329.2114	35.8636	12.1132
32	431.5789	63.4620	21.5321
64	299.0517	76.3349	24.1314
128	301.3376	73.6201	23.1412
256	275.9084	72.6774	25.3414

Use the results in the above table to make the following graph:



Question 4 – Explain your results from Question 3.

As we can see from the graph above, small window sizes produce low throughput, because the pipeline is not utilized efficiently – very few packets are in transition at each time and both sender and the receiver spend a lot of time just waiting for incoming messages. However, as the window size grows, the pipeline is better filled with packets and a higher throughput is achieved with the 10ms delay connection peaking at the window size of 32. Based purely on the connection statistic, he bandwidth-delay product calculations show that the most optimal window size for the 10ms one-way delay is the window size of 24 packets ((10MBits * 20ms RTT)/(1kB)), 240 packets for 100ms delay and 720 packets for 300ms delay connection. Naturally, the decline in the throughput after moving to the window size of 64 for the connection with delay 10ms could be explained by overshooting the most optimal window size value.

Question 5 – Experiments with Selective Repeat

	Throughput (Kilobytes per second)
Window Size	**Delay = 100ms**
8	19.4211
16	37.1941
32	66.9234
64	129.1281
128	197.7425
256	263.2943

Question 6 - Compare the throughput obtained when using "Selective Repeat" with the corresponding results you got from the "Go Back N" experiment and explain the reasons behind any differences.

In theory, Selective Repeat is a faster protocol than Go Back N, because instead of resending the entire window-size-length of packets, it just resends the packets that were not acknowledged by the sender and the pipeline contains as few possible duplicate packets as possible. As we can see from our results in the 2nd column of the table in Question 3 and the table in Question 5, Selective Repeat proved to be faster in our experiments than Go Back N by having almost 4 times higher throughput (263 kB/s for Selective repeat vs. 72 kB/s for Go Back N with the window size of 256).

Question 7 – Experiments with *iperf*

	Throughput (Kilobytes per second)
Window Size (KB)	**Delay = 100ms**
8	24.5
16	42.7
32	49.4
64	55.5
128	55.8
256	59.0

Question 8 - Compare the throughput obtained when using "Selective Repeat" and "Go Back N" with the corresponding results you got from the *iperf* experiment and explain the reasons behind any differences.

The protocol that *iperf* is using TCP which implements the reliable data transfer in a very similar fashion as the Selective Repeat. It looks like the implementation of the reliable data transfer done by *iperf* is a tiny bit more efficient than the Selective Repeat implementation that was written by me. However, when trying to run *iperf* by setting the window size above 64 kB, the program fails to set the window size to the appropriate value and instead sets it to 110 kB. This happens because the maximum window size specified by the TCP protocol is 64 kB and this can not be easily changed. On the other hand, by implementing Selective Repeat in UDP we are not constrained by this limitation. This explains why the throughput values for the *iperf* stop increasing and the Selective Repeat implementation that I wrote achieves significantly higher throughput.