Computer Communications and Networks (COMN) Course, Spring 2014

Coursework: Results Sheet

Forename and Surname:	Mark Nemec
Matriculation Number:	s1140740

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	6895	23.369991
20	2799	22.707713
30	2476	21.905512
40	645	20.830888
50	486	19.795568
60	470	18.935993
70	461	18.246347
80	478	17.284385
90	505	16.378185
100	514	15.608072

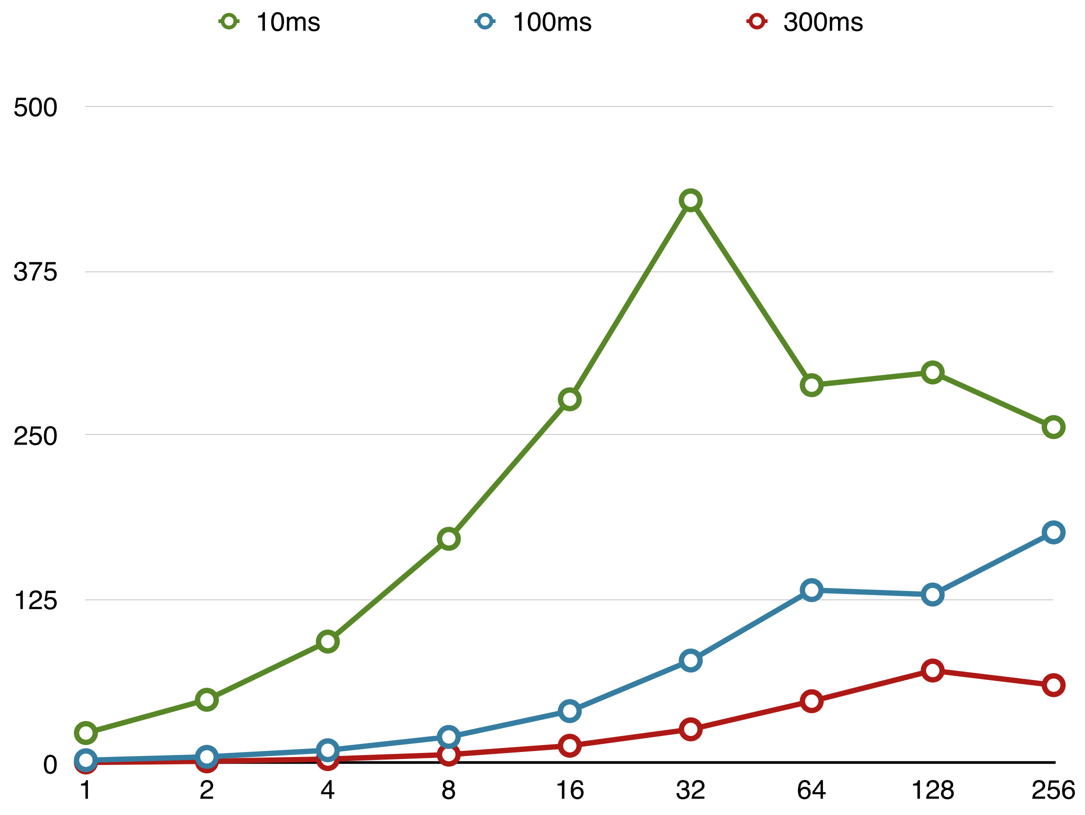
Question 2 – How does the throughput behave when the retransmission timeout increases and why? What is the optimal value for retransmission timeout?

The throughput gets smaller when the retransmission timeout increases. This is because the socket is allowed to wait longer for the incoming acknowledgement packet and thus spends more time being idle and waiting - not sending data.
The optimal value for retransmission timeout is 40. This is when we encountered the sharpest drop in the number of re-transmissions, which we try to minimise. At the same time, however, we want to maximise the throughput. Thus 40 is the ideal candidate. It is also 2 * RTT (Round Trip Time) = 2 * 20ms which is often recommended as the retransmission timeout.

Question 3 – Experiments with Go-Back-N:

	Throughput (Kilobytes per second)		
Window Size	**Delay = 10ms**	**Delay = 100ms**	**Delay = 300ms**
Optimal Retransmission timeout(ms)	40	40	40
1	23.222183	2.532829	0.850040
2	48.516702	5.093422	1.704865
4	92.801511	10.124135	3.409296
8	170.955718	20.249441	6.791328
16	277.070053	39.894214	13.554779
32	428.435537	78.368260	26.034995
64	287.815678	131.938120	47.433001
128	297.519620	128.546395	70.756384
256	255.933585	175.851201	59.635716

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



Question 4 – Explain your results from Question 3.

We can clearly see that the throughput is maximised when we used the lowest delay (10ms). Intuitively, this makes sense because less time is spend idly waiting for a packet to timeout and the file transmits faster.

Moreover, it seems that the window size of 32 has the best performance for the 10ms delay throughput. This could be explained by the fact that having a large window size could have negative effects on performance if the delay is small enough. With large window size we could send all the packets available in the window and then find ourselves needing to resend most of them again if the receiver does not receive one of the first few packets.

On the other hand, large window sizes are useful for transfers with large delays as the risk of losing one packet in many is worth it due to the large delay. This can be seen from the increasing lines of the 100ms and 300ms delay throughputs.

Question 5 – Experiments with Selective Repeat

	Throughput (Kilobytes per second)
Window Size	**Delay = 100ms**
8	102.102997
16	182.259644
32	160.721556
64	154.184918
128	151.811086
256	143.595027

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.

Selective Repeat performs better than Go Back N because the receiver is acknowledging all the packets it receives instead of just the ones received in-order. There is therefore no need to re-send even packets that were received correctly like in Go Back N.

There also does not seem to be much correlation between the window size and the throughput in the Selective repeat algorithm but I would like to point out that the results for this question varied wildly depending on what DiCE machine I used. I got a similar looking result more than once which is why I decided to use it.

Question 7 – Experiments with *iperf*

	Throughput (Kilobytes per second)
Window Size (KB)	**Delay = 100ms**
8	27.6
16	38.4
32	55.5
64	53.0
128	50.8
256	58.9

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.

iperf uses TCP for transmitting data. It is therefore very similar to the selective repeat algorithm. I expected iperf to perform better than my algorithm but it was not the case. I can only assume that this is because it has not chosen the optimal retransmission timeout of 40ms. iperf seems to perform very similarly to Go Back N in the small window sizes but as soon as we get to the window size of 32 we start seeing differences in the throughput. Selective repeat outperforms both in most cases except for Go Back N with a large window but as stated before these results varied depending on the machine.