Computer Communications and Networks (COMN) Course, Spring 2012

Coursework: Results Sheet

Forename and Surname:	Scott Hofman
Matriculation #:	0943941

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

Retransmission timeout (ms)	Number of re-transmissions	Throughput
10	6502	22.779
20	2673	21.459
30	2396	20.433
40	646	20.183
50	642	19.129
60	607	18.588
70	619	17.735
80	597	17.236
90	639	16.251
100	569	16.289

These results use a timeout of 40ms each, with the configurations of

```
ipfw pipe 100 config delay 10ms plr 0.05 bw 10Mbits/s
ipfw pipe 200 config delay 10 ms plr 0.05 bw 10Mbits/s
```

This was due to instructions in the handout which specified that

> "For example, to create a symmetric 1Mbps emulated link with 100ms one-way propagation delay and 1% packet loss rate, you create two dummynet pipes for each direction and configure them as follows (as root):

```
% ipfw add pipe 100 in
% ipfw add pipe 200 out
% ipfw pipe 100 config delay 100ms plr 0.005 bw 1Mbits/s
% ipfw pipe 200 config delay 100ms plr 0.005 bw 1Mbits/s
```

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

The throughput decreases as the number of retransmissions increases. This is a direct result of the length of time – because we must wait for a response before proceeding, packets that are lost have a larger impact on the overall runtime of the program. We must wait longer to confirm that the packet has been lost. The optimal retransmission timeout is the balance between number of retransmissions and throughput. At 40ms, the number of retransmissions drops drastically, but the throughput only dips slightly. This tradeoff simultaneously seeks to maximize the throughput and minimize the retransmissions. To prevent the excessive amount of retransmissions and achieve a high throughput, a 40ms timeout is optimal.

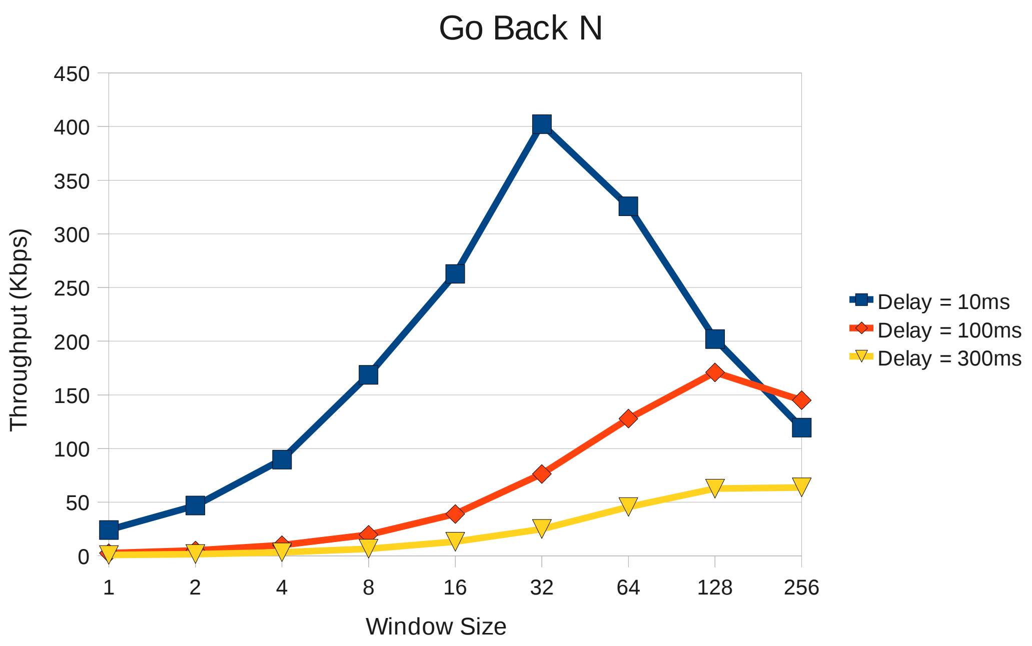
Question 3 – Experiments with Go-Back-N:

	Throughput (Kilobytes per second)		
Window Size	**Delay = 10ms**	**Delay = 100ms**	**Delay = 300ms**
1	24.035	2.481	0.830
2	46.901	4.961	1.658
4	89.668	9.895	3.301
8	168.549	19.684	6.615
16	262.698	38.956	13.113
32	402.316	76.215	24.978
64	325.628	127.77	45.511
128	201.988	171.009	62.665
256	119.327	144.96	63.780

These results use a timeout of 40ms each, with the configurations of

```
ipfw pipe 100 config delay 10ms plr 0.005 bw 10Mbits/s
ipfw pipe 200 config delay 10 ms plr 0.005 bw 10Mbits/s
ipfw pipe 100 config delay 100ms plr 0.005 bw 10Mbits/s
ipfw pipe 200 config delay 100 ms plr 0.005 bw 10Mbits/s
ipfw pipe 100 config delay 300ms plr 0.005 bw 10Mbits/s
ipfw pipe 200 config delay 300 ms plr 0.005 bw 10Mbits/s
```

The results in the above table make the following graph:



Question 4 – Explain your results from Question 3.

Regardless of the delay, the results are similar to linear growth. Each value is approximately double the previous value until it reaches a certain window size. At this point (32 for 10ms, 128 for 100ms and undetermined for 300ms), the window size has become too great and the probability that a packet is lost has become quite high. Since Go Back N is forced to resend the entire window whenever a packet is lost, a higher probability in packet loss results in a higher probability for delay. Increasing the window size beyond this point only increases the chances of resending. We want a window size that utilizes the entire link, but also limits the amount of retransmissions.

With a delay of 10ms and a retransmission timeout of 40ms, each packet has the chance to arrive at the Receiver and return before the transmission timeout occurs. Because of this, its throughput is massive compared to the delays of 100ms and 300ms (each with timeout of 40ms). But a delay of 10ms is unlike something we will see in a real world scenario, hence the reason in the later questions, we do not compare them.

Question 5 – Experiments with Selective Repeat

	Throughput (Kilobytes per second)
Window Size	**Delay = 100ms**
8	19.755
16	39.105
32	76.522
64	131.664
128	160.46
256	191.078

As with above, the resend timeout is 40ms and the configurations are:

```
ipfw pipe 100 config delay 100ms plr 0.005 bw 10Mbits/s
ipfw pipe 200 config delay 100ms plr 0.005 bw 10Mbits/s
```

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.

Comparing only the tables where the delays are the same, we find two different trends. For “Go Back N”, our peak throughput is at a window size of 128; for “Selective Repeat”, our top throughput is at a window size of 256. The trend for Selective Repeat is similar to Go Back N – after a certain threshold, the speed at which the throughput increases relative to window size diminishes. However, rather than dropping down after like Go Back N, Selective Repeat approaches a maximum value. However, increasing the window size will increase the size of the buffer on the Receiver end – too large and the Receiver will have trouble dealing with the packets. This behavior stems from the individualistic behavior of the timeouts – the sender does not have to wait if an individual packet is loss unless it is the first in the window. Without having to wait, having a larger window size does not hinder the application – it increases its effeciency and therefore its speed.

Question 7 – Experiments with iperf

	Throughput (Kilobytes per second)
Window Size (KB)	Delay = 100ms
8	18.41
16	30.0
32	35.12
64	58.75
128	71.87
256	76.5

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.

Comparing the 100ms of iperf with both Selective Repeat and Go Back N, we see that UDP is faster than TCP. This is no surprise – TCP has an overhead in both setting up the connection and processing that slows it, but this makes it more reliable. In terms of growth, iperf performs more closely with Selective Repeat – there seems to be a linear growth. However, the results provided here are the result of several tests – the variability of TCPs speed prevented a clearly defined result. On different machines, these tests produced differing results.