Project2: Simple FTP with Go-back-N

Qiufeng Yu (qyu4

Mingxuan Shi (mshi4)

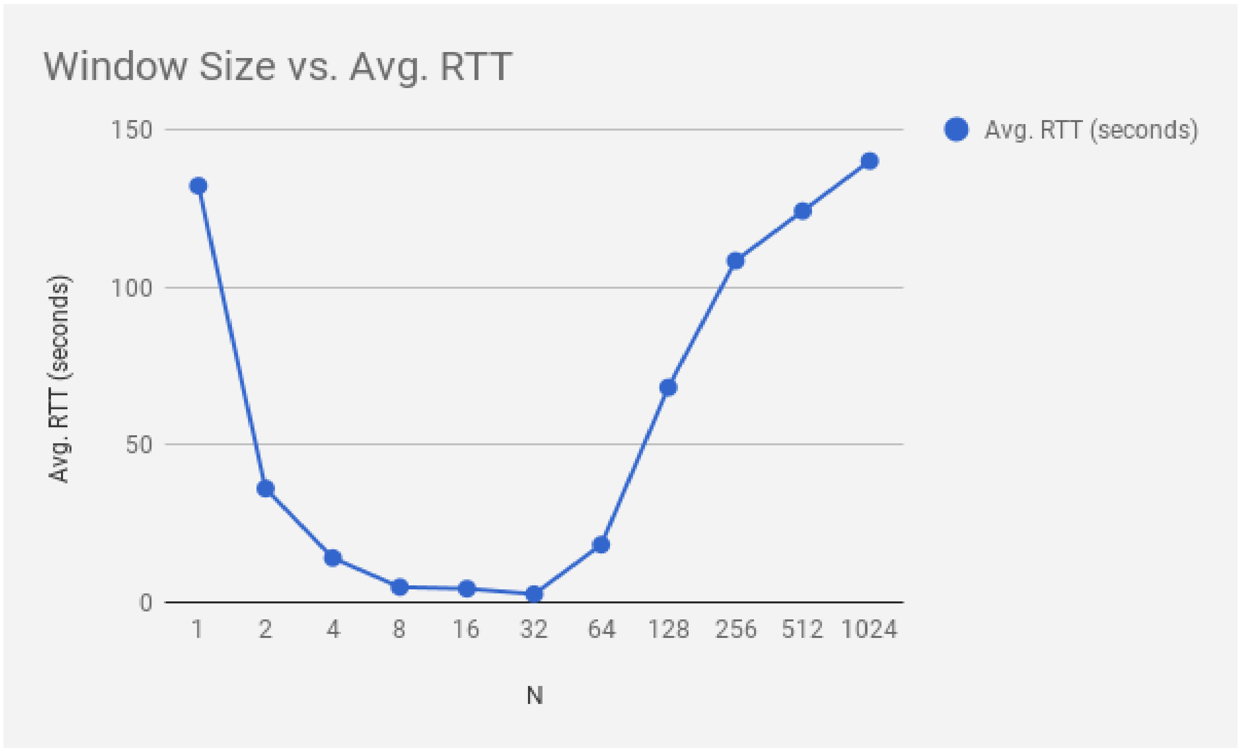
In this project, we have implemented a simple FTP protocol using UDP to transfer a 1MB file from the client to the server using the Go-Back-N scheme. We performed the following three experiments to evaluate the effect of the window size N, maximum segment size MSS, and packet loss probability p, and the relationships between them and the total round trip delay for transferring the 1MB data file. For this particular project, we performed the experiment on two machines, one local laptop, and the other one is an Ubuntu 16.04 EC2 instance on Amazon AWS. These two machines were separated by 9 hops and the RTT between the two machines was 3.6ms. We measured the RTT five times for each task and put the average RTT into the result.

TASK1: Effect of Window Size N

Fixed variables: MSS = 500, P = 0.05

Results:

N	Avg. RTT (seconds)
1	132.125
2	36.1527
4	14.1245
8	4.859
16	4.356
32	2.645
64	18.326
128	68.265
256	108.326
512	124.124
1024	140.021



Explanation:

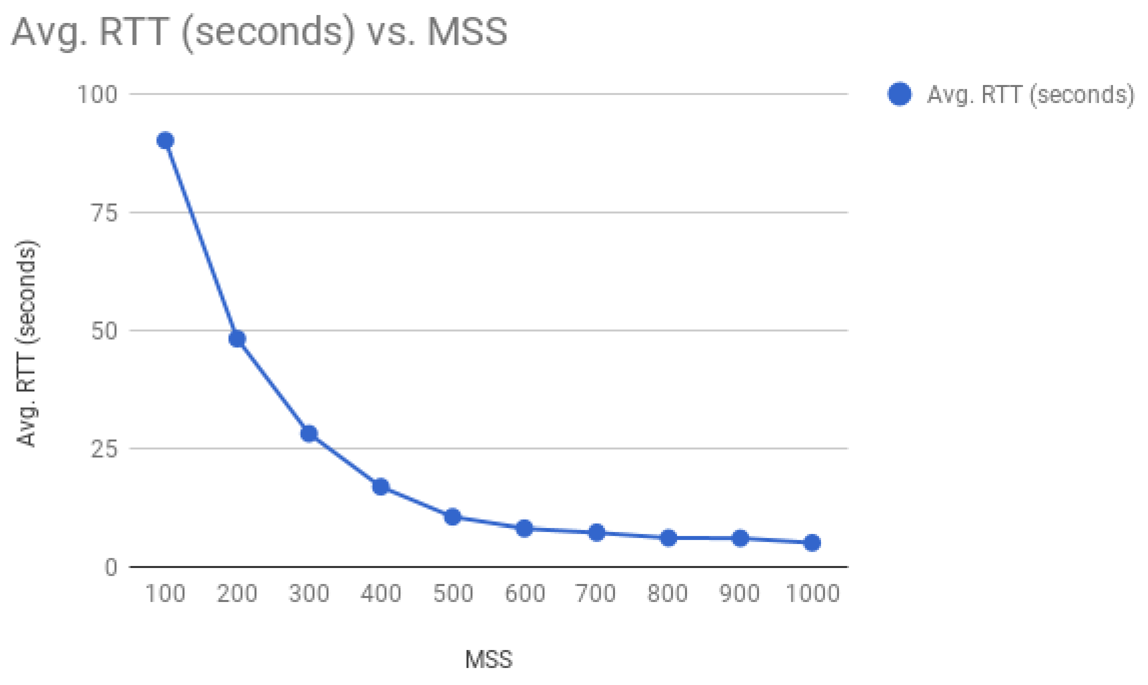
For this task, when N=1, it's basically the Stop-and-Wait protocol, in this case, given that the timeout counter is 500ms, the receiver (server) will only return 1 ACK for each N. In other words, client sends 1 packet at a time and wait for ACK, until sending the next packet, so it's very slow. As the window size increases, throughput increases, therefore, RTT decreases. When N = 32, we observe that we have the least RTT. On the other hand, when the window size N is large, when a packet is lost or an ACK is lost, the whole packets that haven't been ACK'd have to be sent again, which leads to higher RTT, that's why when N > 64, we observe an increase in the RTT.

TAKS2: Effect of MSS

Fixed variables: N = 64, p = 0.05

Results:

MSS	Avg. RTT (seconds)
100	90.154
200	48.254
300	28.145
400	16.897
500	10.521
600	8.125
700	7.258
800	6.154
900	6.098
1000	5.124



Explanation:

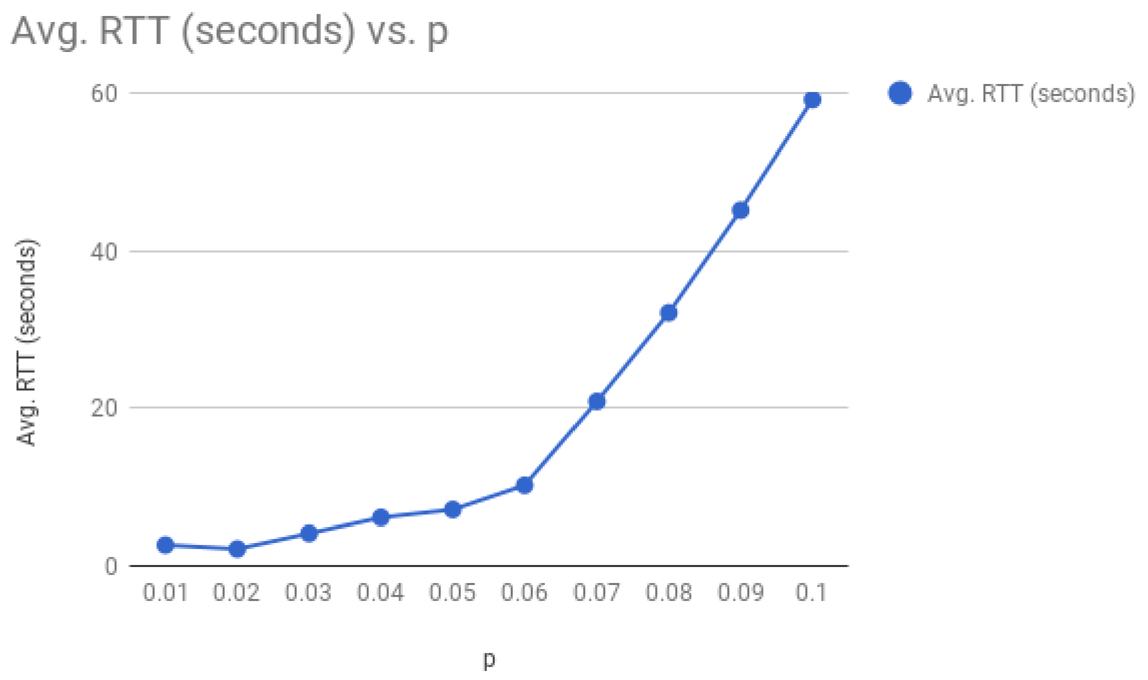
We can see from the graph above, RTT decreases as MSS increases exponentionally. Reason being is that when MSS is very small, there are more number of packets need to be sent than situations when MSS is large. Due to timeout or loss of ACK, more number of packets have to be sent, hence leads to large RTT. However, as MSS gets larger, less packets need to be transmitted, which leads to less RTT time. One thing to notice is that, high MSS works well only when we have more sophisticated network bandwidth, otherwise, even though MSS is large, poor network bandwidth can be a bottleneck.

TAKS3: Effect of Loss Probability p

Fixed variables: N = 64, MSS = 500

Results:

p	**Avg. RTT (seconds)**
0.01	2.658
0.02	2.154
0.03	4.125
0.04	6.125
0.05	7.124
0.06	10.197
0.07	20.859
0.08	32.124
0.09	45.158
0.10	59.159



Explanation:

According to the data and the graph above, we can conclude that as p increases, RTT increases. This result is not surprising because when p increases, we manually make more packets lost, which leads to retransmissions on the client side. Therefore, total round trip time increases as p gets larger.