Computer Networks **Shreshth Tuli, Samarth Aggarwal**
COL334 - Laboratory Assignment 2 2016CS10680, 2016CS10395
Supervisor - Prof. B.N. Jain Date: 26/09/2018

This Lab Assignment aims to implement and analyze Go Back N - Data Link Layer protocol.

1 Introduction

Go-Back-N is a type of sliding window protocol wherein the sender continues to send frames even before receiving an Ack, up to a limit specified by window size. The window size on the receiving end is necessarily zero. Upon completion, we expect to have a working network model that implements a 2-host-2-switch network topology and exchanges data using the aforementioned protocol. To be more precise, the inference from this model should not only indicate the direction of changes in the network performance when a parameter (say window size) is changed, but should also identify the extent of changes in the performance. This is precisely why, a real-time simulation of the protocol is required. In order to measure the performance of any system, certain evaluation metrics have to be defined. For this sake, we will measure the performance of our system on the basis of throughput and latency. It is to be noted that these two performance measures are not directly observed from the network realization. Rather certain other parameters have to be observed from which these are calculated. To calculate throughput, we need the number of bytes of data transferred in a given time. To calculate latency, we need the number of packets successfully transferred over the network in a given time. Also, these observations must be made over large data samples so as to minimize random error.

2 Implementation

To realize the above network protocol, we have used Mininet. The 2-host-2-switch network topology is coded in python and is given to Mininet to implement. Mininet allows us to set link parameters like bandwidth, delay, loss, etc. We use Mininet to limit bandwidth to 1 Mbps and set delays. Packet loss percentage we set at the Networking entity - 'host.py'. Two hosts are initialized by Mininet and we then run the program 'host.py' on both the hosts. This program first assigns ports to both the hosts for sending and receiving data. The port for sending data for host 1 is same as the port for receiving data for host 2 and vice-versa. For each of the host, the first step taken by the data-link layer is to construct frames out of the packets that it is receiving from the network layer. A single packet may be broken into multiple frames. Apart from the data of the packet, a frame also contains header that includes the sequence number of the frame and a byte to denote whether this frame is a data frame or an Ack frame. Even before the frames are sent, a receiver thread is initiated to listen to the data and Ack framed coming from the other host. Next, frames are sent down to the physical layer (Mininet in this case) to pass them to the other host. As many frames are sent as allowed by window size. After this, the sender waits till it receives the ack to the first frame of the sliding window. In case of timeout, it re-transmits the frames. As soon as it receives this ack, it shifts the sliding window by a unit allowing a new frame to transmitted.

Note that receiving has to be done by a host on another thread since both sending of data frames and receiving them happen simultaneously. A mutex lock has been used due to presence of threads. Some helper functions have also been implemented to achieve the above functionalities. These include timer to achieve the start, stop, reset functions of the timer. The packet module makes a frame from packet data and extract data from frame to make packet again. The packet drop rate is simulated by generating a random number between 0 and 100. If the random number generated is less than the drop rate, then the packet is dropped else it is transmitted.

3 Observations

The performance of the network on tweaking error probability, window size and delay, has been summarized in a table. The testing platform had the following characteristics:

1. Mininet using Virtual Box VM on Dell XPS 13 Laptop (Dual Core i5, 8 GB RAM)
2. Size of file used for the experiment = 927,763 bytes
3. Controller : Remote controller on localhost

Error Probability	Window Size	Delay (in ms)	Number of Timeouts	Total Time	Number of Packets Transferred	Throughput (in KBps)	Latency (in ms)
0.01	5	0	0	74.181	7304	12.50674701	10.15621577
		3	0	75.715	7282	12.25335799	10.39755562
		5	0	76.087	7211	12.1934496	10.55151851
	7	0	0	54.826	7382	16.92195309	7.426984557
		3	0	54.589	7305	16.99542032	7.472826831
		5	0	54.461	7272	17.03536476	7.489136414
	9	0	0	42.748	7372	21.70307383	5.798697775
		3	0	43.015	7316	21.56835987	5.879579005
		5	0	44.147	7298	21.01531248	6.049191559
0.05	5	0	264	108.219	7116	8.573013981	15.20784148
		3	277	111.381	7356	8.329634318	15.14151713
		5	287	112.173	6979	8.270822747	16.07293308
	7	0	275	92.961	7382	9.980131453	12.59292875
		3	296	92.088	7141	10.07474372	12.89567287
		5	291	92.081	6940	10.07550961	13.26815562
	9	0	267	77.118	7442	12.03043388	10.36253695
		3	281	78.78	7353	11.77663112	10.71399429
		5	298	81.049	7345	11.44693951	11.03458135
0.1	5	0	674	162.471	7105	5.710329843	22.86713582
		3	715	164.955	7150	5.624339972	23.07062937
		5	723	166.596	7117	5.56893923	23.4081776
	7	0	707	145.738	7292	6.365964951	19.98601207
		3	728	147.537	7249	6.28834123	20.35273831
		5	746	149.056	7167	6.224257997	20.7975443
	9	0	706	128.114	7468	7.2416988	17.1550616
		3	702	128.186	7337	7.237631255	17.4711735
		5	698	130.162	7122	7.127756181	18.27604605

4 Analysis

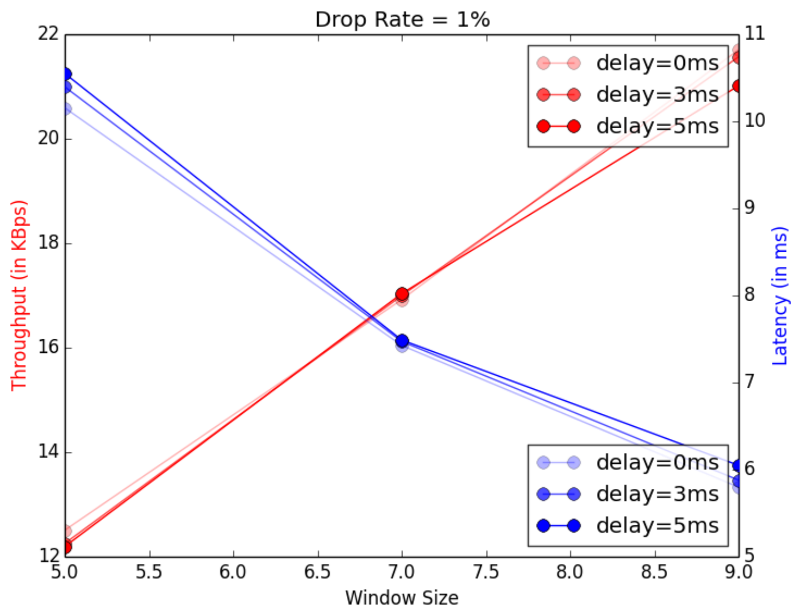

Figure 1: Performance when drop rate is 1%

From the graphs obtained, we observe that:

- As the drop rate of the network increases, the throughput also decreases. This is because when drop rate is high, even when the network attempted to send the same number of packets, the actual number of packets that it was successful in sending was lesser. Due to this, the data transferred by it in unit time decreased.

- As the drop rate of the network increases, the average latency increases. This is because even though a successful packet may require the same amount of time as before to be transferred, the dropped packets consume time but don't add to the number of successful packet transfers. Hence, they reduce the average latency.

- As is clearly the trend observed in the graphs, increase in the window size leads to an increase in throughput. This is because more the window size, more are we waiting before we re-transmit our frame. It may happen that an Ack corresponding to successful receipt of a frame was not received in a short interval of time (corresponding to smaller window size) but was received when the waiting time was increased (corresponding to larger window size). Due to this reason, we can avoid certain re transmissions simply by waiting for the Ack for a longer while thereby increase window size.

- The graphs also suggest that latency decreases with increase in the window size. The reason to this observation is similar to the previous observation. Increasing the window size will lead to fewer futile transmissions of the frames that were perceived to be dropped/unsuccessfully transmitted. Due to this, more frames will be successfully transmitted within the same time interval thereby decreasing the latency.

- The changes in throughput due to change in delay is negligibly small. Even then, throughput is marginally higher in case of no delay and decreases by small amounts as the delays

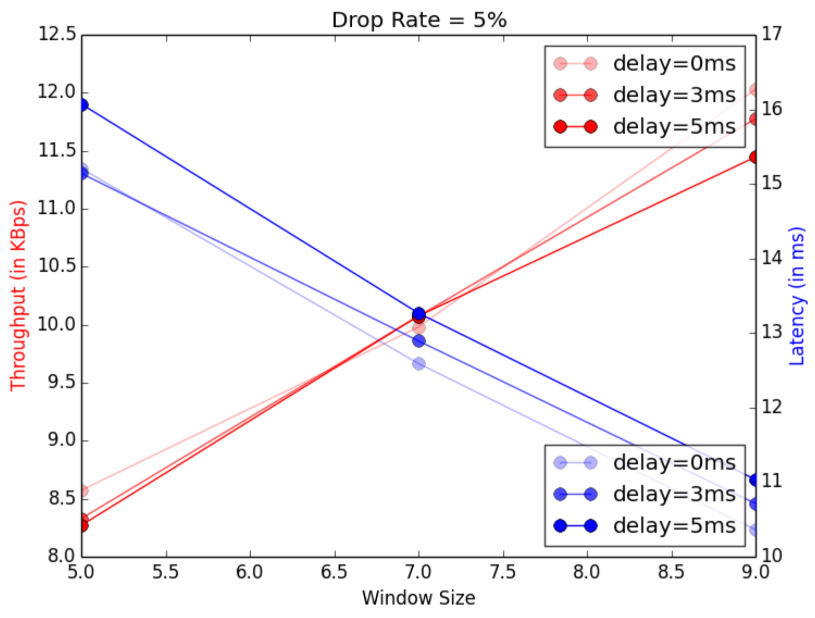

Figure 2: Performance when drop rate is 5%

increase. Also, the difference in throughput is more evident when the window size is large. This is because the delays are accumulated over the window. If the window size is large enough for the accumulated delay to cause timeout, only then will the throughput decrease. In case of low window size, although the delays would have accumulated but even then their sum would not be sufficient to overstep the timer limit.

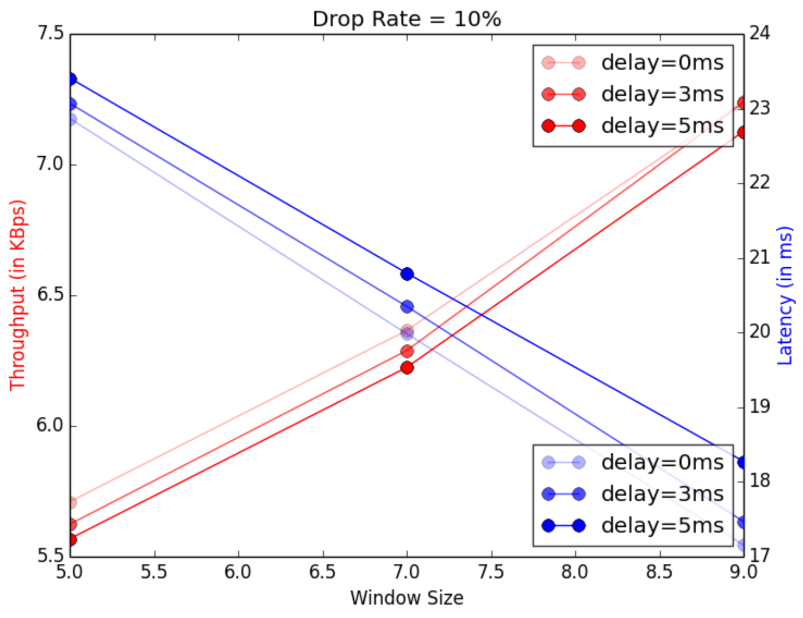

Figure 3: Performance when drop rate is 10%

- An increase in delay leads to an increase in latency. Although, the extent of change is very small even in this case. The results are consistent with theory because if the delay increases, then the total time required to send the same number of packets will increase. Hence, the time required to send a single packet will also increase on the average.

5 Conclusion

- Throughput decreases with increase in drop rate.
- Latency increases with increase in drop rate.
- Throughput increases with increase in window size.
- Latency decreases with increase in window size.
- Throughput decreases with increase in delay.
- Latency increases with increase in delay.

Appendices

- **End-to-end delay** refers to the time taken by a packet to go from source to destination over a network.
- **Throughput** refers to the rate at which data can be successfully transferred over a network.
- **Drop rate** refers to the fraction of packets dropped on an average over the network due to reasons such as erroneous transmission, limited buffer capacity, etc.