Switch Simulation for Performance Measurement

Abhiram Singh, Sidharth Sharma
Course Instructor: Prof. Ashwin Gumaste

1 Introduction

In current network scenario, end to end packet delay is measured as sum of propagation delay at each hop and processing delay at all intermediate switching devices.

Packet processing delay involves packet transmission delay and the time for which packet waits in the switch buffer (also known as queuing delay). Buffers are provided because of the following reasons:

- To compensate the rate mismatch between transmission devices.
- To handle bursty traffic.
- To improve the link utilization.
- To hold the packet until table look-up happen.
- To hold the packet until destination port becomes free for transmission.

Buffers in the switch try to solve above mentioned problems but give rise to some other problems.

Out of mentioned three delays, propagation and transmission delay for a packet is fixed for a given path and transmission rate of each intermediate switch port. However, queuing delay cannot be measured exactly for each packet because of the following reasons:

- Due to bursty traffic flow.
- Due to behaviour of other flows is unknown.

Since queuing delay is the key deciding factor in providing Quality of Service in the network. Hence, it is necessary to observe its behaviour.

2 Problem Description

To observe the switch performance (throughput and port to port delay) with varying packet sizes.

3 Simulation Details

For switch simulation, we used *Simpy* discrete event simulation library in *Python* .
Switch of 16 ports each of 10Gbps capacity is implemented.
Parameters of packets are defined in the class *Packet*.

For traffic generation, *PacketGenerator* class is defined which generates packets according to given packet size and generation rate.

Corresponding to each input port, input buffer is defined. Packet waits in the input buffer until it finds its output port in the switch based on the look-up. *Port* class defines the functionality of input buffer.

For each input buffer, there are 15 Virtual Output Queues(VoQ) corresponding to each output port. Packet stays in the VoQ until the transmission of all the front packets in the VoQ is completed. This wait depends on how many packets are in front of VoQ and how much contention each packet is experiencing from the VoQs associated with other input ports. Code for VoQ is defined in the class *VOQ*.

We attach a packet sink with the output ports in which all the packets are collected. This functionality is defined in the class *PacketSink*. To measure switch throughput, we defined *SinkMonitor* class which periodically checks packet sink and update the throughput.

4 Results

We measured the throughput and port to port latency in the switch by varying packet sizes.

Delay vs packet size graph shows that on increasing packet size, port to port delay of the switch decreases. Reason behind this is more look-ups are required for smaller size packets in contrast to the larger size packets (less number of larger size packets are generated for the same data rate). This result is shown in Figure 1.

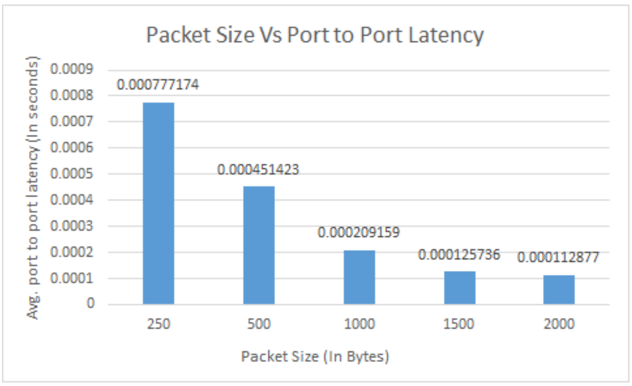

Figure 1: Graph showing decrease in the port to port latency on increasing packet size.

On increasing average packet size we observed increase in throughput because port to port latency per packet for larger packet size is less. This result is shown in Figure 2.

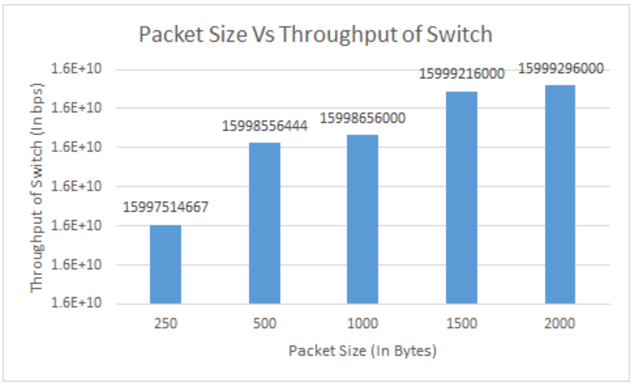

Figure 2: Graph showing increase in the switch throughput on increasing packet size.

5 Conclusion

The actual capacity of the switch varies with the average packet size of the flow. We observed the dependency of throughput and port to port latency of packet on average packet size of the flow. Therefore, switch can be utilized in an efficient manner if larger average packet sizes are used in the flows.