Abhinav Kuruvadi
November 5, 2014

Project 4 - Report

Load generation process

In this project, I am generating load on every "toc" and randomizing the set of output destination computers. Since the writeup mentions that a computer is not allowed to generate more than one packet every cycle, all the computers generate a packet destined to a random destination computer which is calculated every toc. This will ensure that over a large period of time (Around 10000 tocs), the distribution is well dispersed and the can provide the results.

Input Queue Priority

To ensure that the input queues are given equal priority, on every toc, I start iterating from the next queue and over a large no. of tocs, all the queues would have received the same preference.

Parameters:

1) **Switch Size :** Controlled by changing the number of computers which will send packets to destined buffers connected to the switch.

2) **Delay:** Packet delay is added to all the packets on the input buffer of the switch every toc.

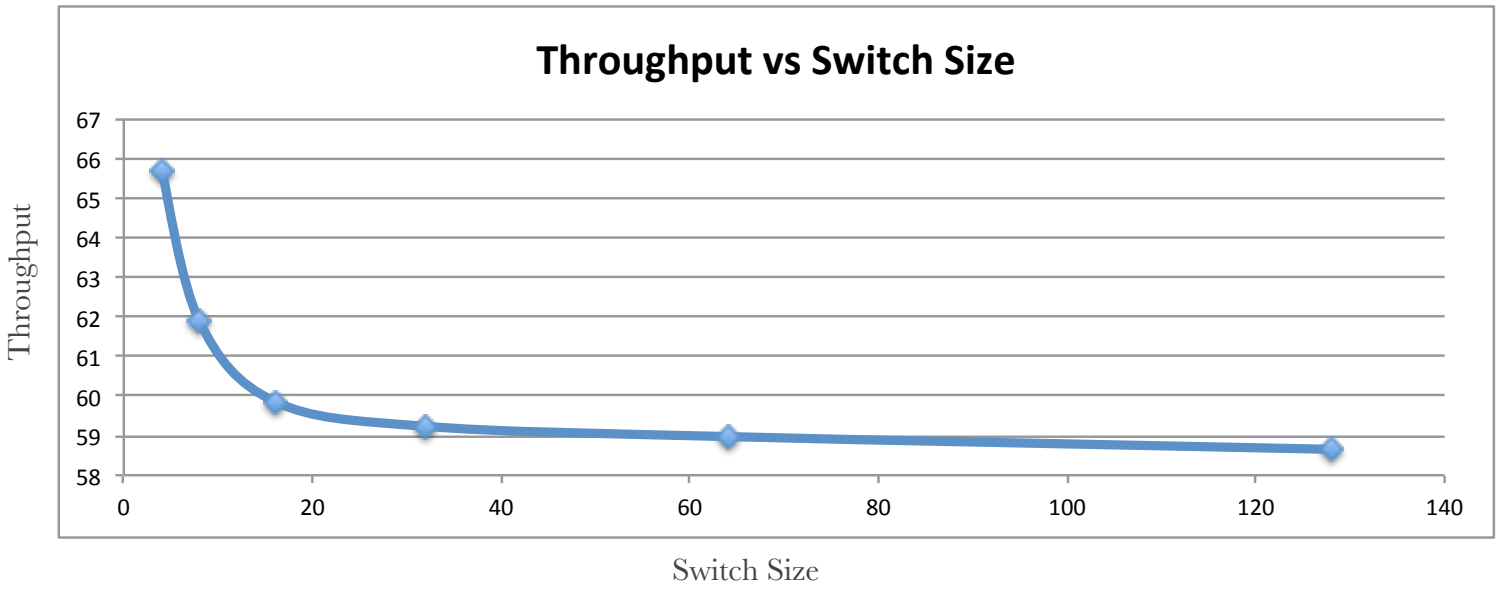
3) **Buffer Length:** Controlled by changing the bufferSize variable in the NIC class.

4) **LookupLevel:** The level upto which the switch searches in the input buffer in case the head of the line is blocked.

5) **SpeedUp:** The memory speedup that the switch can have. This number equates to the no of input buffers that can simultaneously write into the same output buffer.

Input Queuing

This was the behavior observed when the switch size was gradually increased. Throughput tends to stabilize at 58.6%. This occurs due to head-of-line blocking issue where one of the input buffers is unable to send the packets along to the output buffer as it is blocked by another input buffer. the output buffer is set to be of size 1.

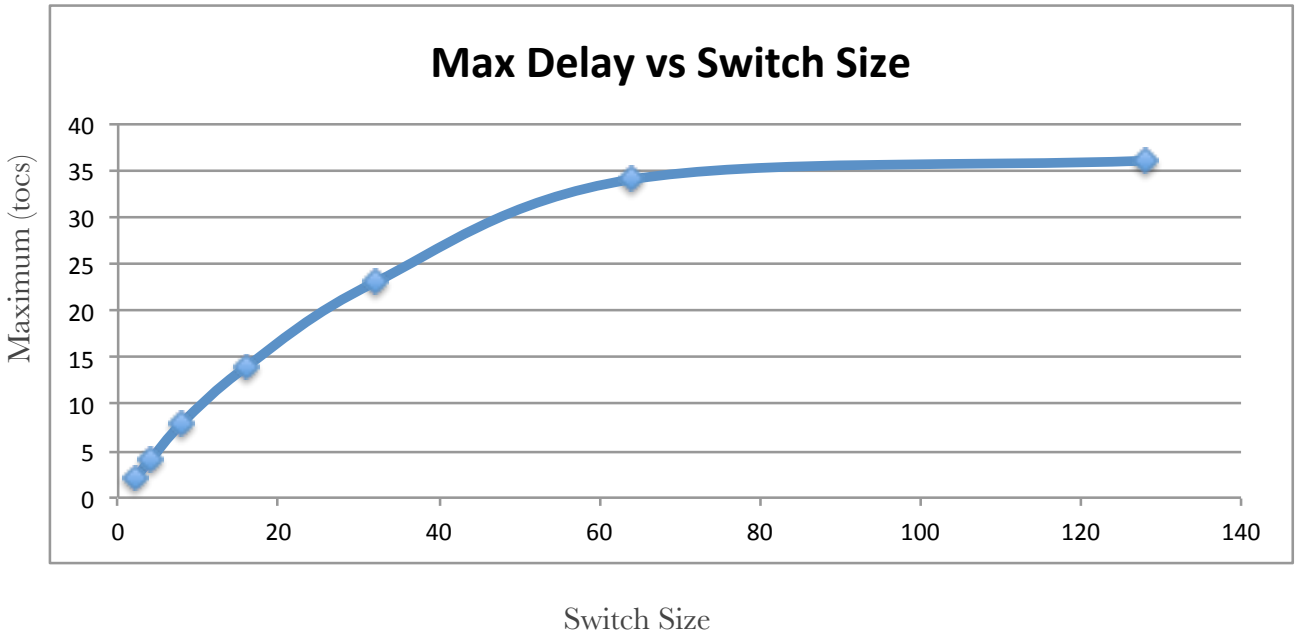


Two methods were implemented to overcome this issue:

1) Speedup(S) : As explained in the paper "Matching Output Queueing with a Combined Input Output Queued Switch", this technique implies that the switch is capable of removing upto S packets on the input and deliver it to the output every time period.

2) LookupLevel: As explained by the professor in class, when the head of the line blocking occurs, I am looking at the packets behind the head if they have a free channel to reach the destination. Effectively avoiding the HOL issue.

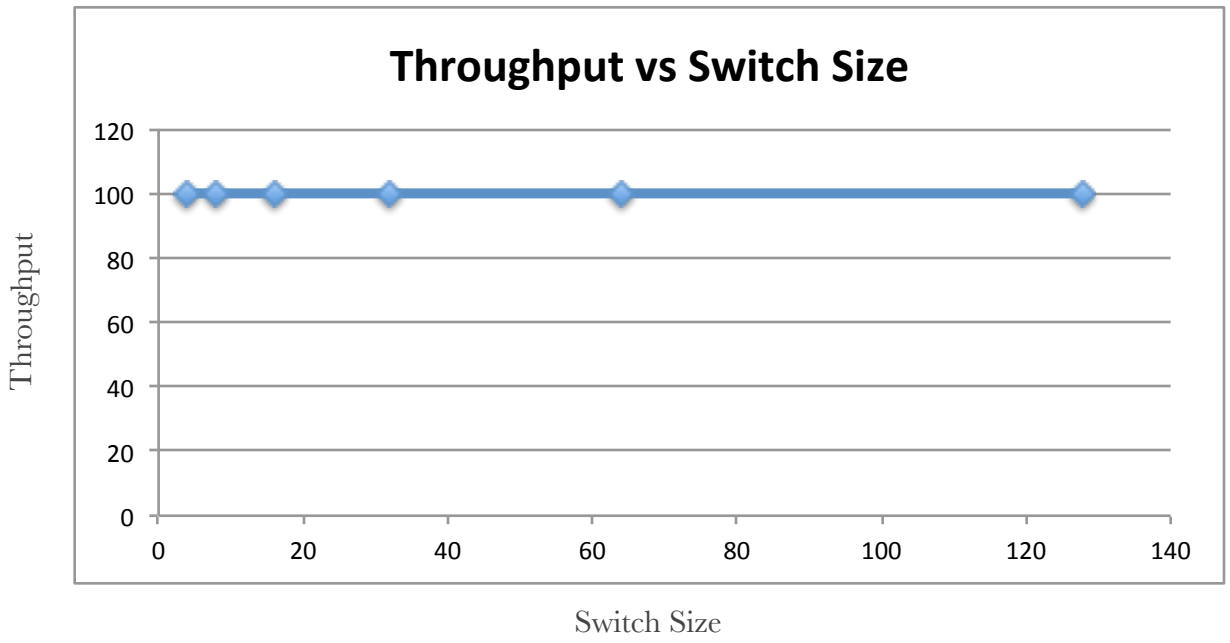
Input queuing : Maximum delay vs Switch Size:



The Y-Axis is the value of the maximum delay experienced by any cell during the run. It stabilized around 36 after increasing the switch size to 128. Each run was carried out for 10000 tocs.

Output Queueing

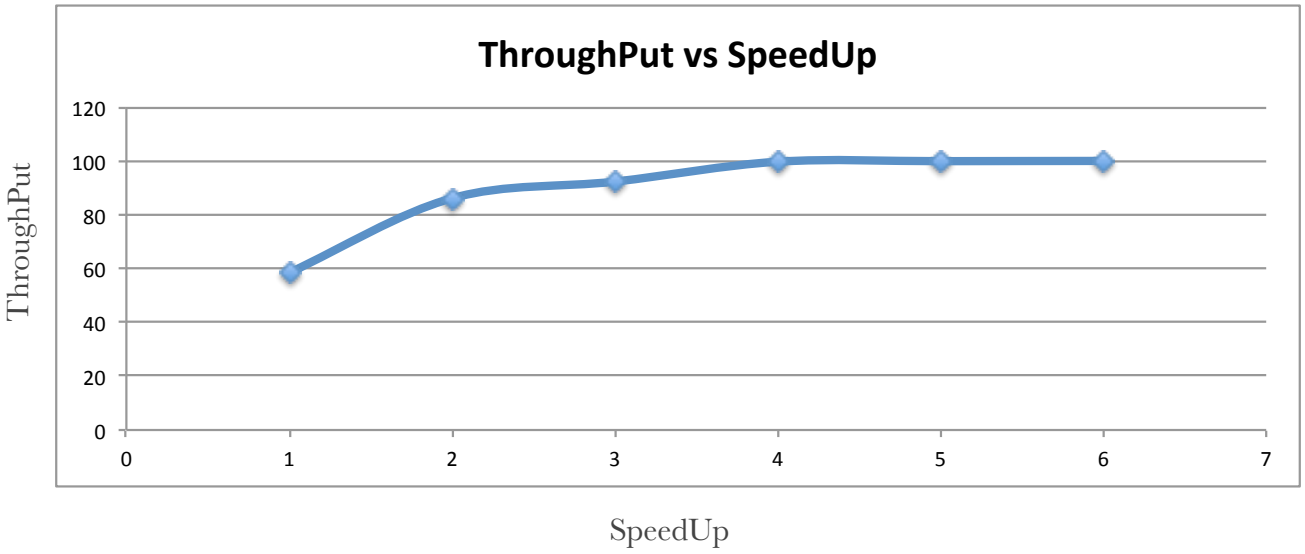
Since output queueing works with a speed up on n compared to input queuing techniques, It can be observed that the throughput is 100% for all switch sizes. The input buffer here is set to be of size 1. This is a very expensive technique and moreover, for high speed switches, there is no memory available that can provide this bandwidth.



Speed-up technique is a possible approach that could be much cheaper compared to the output queuing technique. In this technique, instead of n, the speedup was increased gradually from 1 and the throughput was recorded for each speedup.

Input and Output Queuing

I have introduced a new Input and Output queuing mode in the switch to test the speedup technique. The speedup can be controlled by changing a member variable in the Switch class. The output buffer size is set as the selected speedup value.

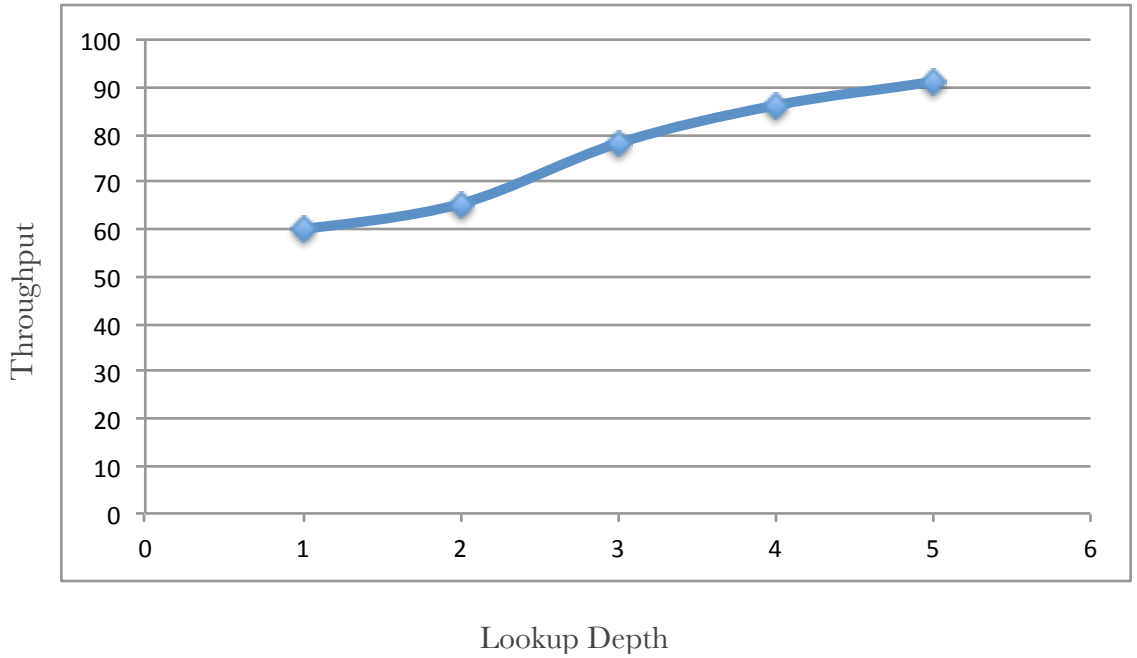


This experiment was carried out with a switch size of 32 and the input buffer size of 20.

As it can be observed from the graph, a speedup of around 4 is sufficient to reach a throughput of 99.7%. This could be an effective solution to the head of line blocking issue. A speedup of 6 showed a staggering 99.97% throughput which is really close to the ideal n-speedup throughput.

Lookup Level Technique

As explained by the professor in one of the lectures, when the switch realizes that the buffer is blocked by the head of line packet, it can look deeper into the buffer and see if there are any other packets destined to a possible free output channel.



As it can be seen from the graph, with a lookup of depth 5, I observed a throughput of around 91.5%. By increasing further, it stayed at around 92%.

(This code was implemented a week ago and I broke in the last minutes after implementing the speedUp technique.)