CS6040: Router Architecture and Algorithms

Semester: Jul-Nov 2024

Assignment 3 Report

Packet Switch Queueing

Name: Snehadeep Gayen
Roll: CS21B078

Submission Date: 6th Oct 2024

1 Introduction: Packet Scheduling in Switches

Packet switches and routers use a switched backplane based on simple 2x2 crossbar switches. Most often, these systems use input queues to hold incoming packets waiting to get switched through the switching fabric to their destined output ports. The packet scheduling & queueing algorithm used for this can greatly affect the performance of the switch. This assignment explores the popular iSLIP algorithm and compares them to classical No-Queuing (NOQ) and Input Queueing with Head-of-Line Blocking cases.

2 Challenges

Input Queueing with a simple First-In First-Out scheduling is known to suffer from Head-of-Line blocking effect in which, a blockage at the HOL make all the subsequent waiting packets in the queue suffer. Further, while guaranteeing high utilisation and low drop probability, the scheduling algorithm must also respect flow-wise packet order, guarantee fairness and must adapt quickly to changes in the traffic or bursts in the network. All these challenges, make it difficult to design scheduling algorithms for packet switches.

3 Simulation Experiment

This experiment designs a switch simulator to test out different scheduling algorithms, and compares their drop probabilities, delays and link utilisation. Further the simulator is flexible to handle many different parameters and is easily extensible to other scheduling algorithms.

4 Variables of the Experiment

4.1 Number of Switch Ports N

The switch is assumed to have N input ports and N output ports, numbered from 0 to $N-1$. The default value is 8.

4.2 Buffer Size B

Each input and output port can hold up to B fixed-length packets. The default value is 10.

4.3 Packet Generation Probability p

Each input port generates a packet in a time slot according to a Bernoulli(p) distribution,

where p denotes the probability that a packet is generated. The default value is 0.5.

4.4 Queue Type q

This argument specifies the type of queue used for packet handling. Possible values include NOQ (No Queuing), INQ (Input Queuing), and CIOQ (Combined Input and Output Queuing). The default queue type is INQ.

4.5 Knockout K

This parameter is specific to the CIOQ queue type and determines the backplane speedup compared to the output link transmission time. $K = 2$ means that in one transmission time unit of the output port, the backplane can switch two rounds of packets. The default value is 1.

4.6 Input Queue Length Lookahead L

This parameter is also specific to the CIOQ queue type and defines the length of the input queue to be considered for packet scheduling. $L = B$ means that all the packets in the input queue will be considered for scheduling. The default value is 1.

4.7 Maximum Slots T

This argument specifies the total number of time slots for the simulation. Each time slot corresponds to the transmission time of one packet. The default value is $10,000$.

5 iSLIP Algorithm [1]

5.1 Algorithm

Algorithm 1 iSLIP Algorithm

```
1:  Initialize: All inputs and outputs are unmatched
2:  repeat
3:      Step 1: Request
4:      for each unmatched input do
5:          for each output with a queued cell do
6:              Send a request to the output
7:          end for
8:      end for
9:      Step 2: Grant
10:     for each unmatched output do
11:         if output received requests then
12:             Grant to the next input in round robin order
13:         end if
14:     end for
15:     Step 3: Accept
16:     for each unmatched input do
17:         if input receives a grant then
18:             Accept the grant in round robin order
19:             Update round robin pointers for both input and output
20:         end if
21:     end for
22: until no change in schedule
```

5.2 Time Complexity

Using various simualations with a $N \times N$ switch, the authors suggest that the iSLIP algorithm converges in $\log_2 N$ steps [**1**]. However, they have not been able to prove the bound $\mathbf{E}(I) \leq log_2 N$. Note that this is a much faster algorithm that classical Maximum Weight Matching scheduling algorithm which requires $\mathcal{O}(N^3)$ time to converge, and has time complexity comparable to Parallel Iterative Matching algorithm which is proved to converge within $\log_2 N + \frac{4}{3}$ iterations on average [**1**] .

5.3 Salient Features

5.4 Example Run

Consider the following example for the iSLIP algorithm. Initial state of all the input queues is shown below. Assume that initially all the round robin (RR) pointer are initialised to 0. Ports are considered to be 0-indexed, that is, after a packet from the ith input port

is accepted to the jth output port, the RR pointer of the ith input port will be set to j and the RR pointer of the jth output port will be set to i. For this example, $N = 4$ and $L = 2$, that is, the first two packets in each port is considered by the algorithm.

Table 1: Initial State of the Switch

Input Port	**Queue State**	**RR Pointer (Input)**	**RR Pointer (Output)**
0	[Packet 1, Packet 2]	0	0
1	[Packet 3, Packet 2]	0	0
2	[Packet 3, Packet 0]	0	0
3	[Packet 0, Packet 1]	0	0

In the first iteration these are the grants received each output port and the acceptances issued by each input port.

Table 2: Grants Received and Acceptances Issued in the First Iteration

Output Port	**Grant Received From**	**Input Port**	**Accepts Received**
0	Input Port 2 and 3	0	Output Port 1 and 2 (1 accepted)
1	Input Port 0 and 3	1	Output port 3 (3 accepted)
2	Input Port 0 and 1	2	Output Port 0 (0 accepted)
3	Input Port 1 and 2	3	None

As we can see, each Output ports receives two grants, for each of the two packets stored among the NL packets in the input queues, destined to that particular output port. Each output port then selects the next packet in RR fashion after the RR pointer and sends a accept request to the corresponding input port. Each input port then selects the next packet in RR fashion again according to its own RR pointers.
Thus after this iteration, the matching of input and output ports is $(0, 1), (1, 3), (2, 0)$. Thus the corresponding RR pointers of the ports are updated as follows:

Table 3: Initial State of the Switch

Input Port	**Queue State**	**RR Pointer (Input)**	**RR Pointer (Output)**
0	[~~Packet 1~~, Packet 2]	1	2
1	[~~Packet 3~~, Packet 2]	3	0
2	[Packet 3, ~~Packet 0~~]	0	0
3	[Packet 0, Packet 1]	0	1

In the next iteration, only the input ports, and the output ports which are not yet matched are considered. The grant table is described below:

Since the output ports which received the grants are already matched, there are no Acceptance issued, and the matching remains unaltered. Thus, the iterations stop and the following mapping of ports is output.

Table 4: Grants Received and Acceptances Issued in the First Iteration

Output Port	Grant Received From	Input Port	Accepts Received
0	Input Port 3	0	None
1	Input Port 3	1	None
2	None	2	None
3	None	3	None

Table 5: Final Mapping of Input and Output Ports

Input Port	Output Port
0	1
1	3
2	0
3	None

6 Results

6.1 Comparison with different probabilities

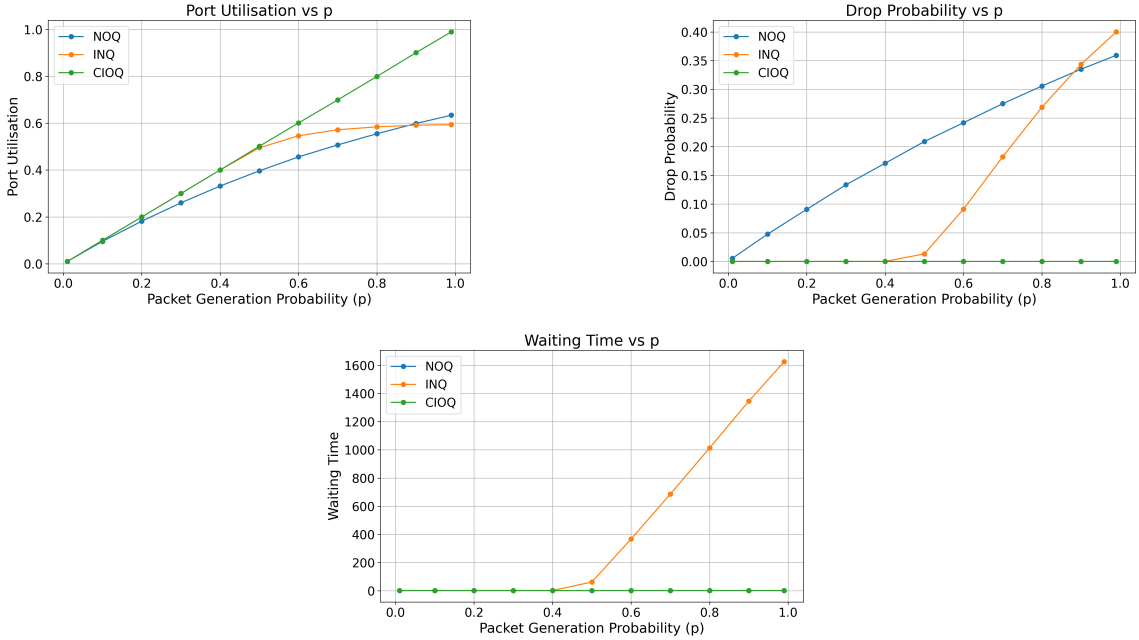


Figure 1: Comparing the different Queueing algorithms

The above figure compares the link utilisation, drop probability and waiting time for the different algorithms with default parameters. For CIOQ, $K = 4$ and $L = 6$ was chosen.

6.1.1 Utilisation

From the figure we see that iSLIP algorithm has the best utilisation and increasing linearly with the increase in probability. Also we see that the values for $p = 1$ **agree with**

the theoretical maximum utilisation of 64% **for NOQ and** 58.6% **for INQ**. One important insight from this is that although NOQ outperforms INQ, at lower loads, the INQ gives a much better utilisation than NOQ.

6.1.2 Drop Probability

CIOQ with iSLIP algorithm has as a consistent nearly 0 drop probability, whereas NOQ and IOQ have drop probabilities reaching to upto 40% for $p = 1$. Again, here INQ outperforms NOQ under low loads, but becomes worse under higher loads.

6.1.3 Waiting Time

The waiting time for NOQ and CIOQ is consistently 1 time slot. But for INQ, the wait time increase approximately linearly after $p = 0.5$, and reached a maximum of 1600 time slots. Thus, considering waiting time as the metric, NOQ and CIOQ are preferred.

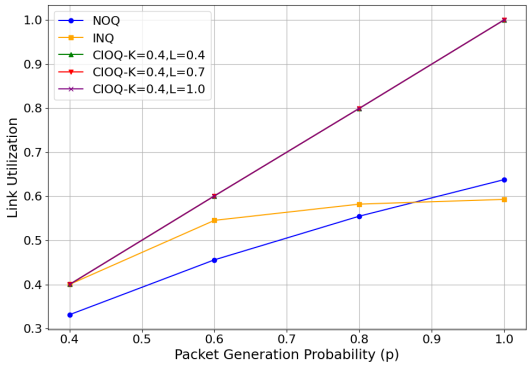
6.2 Performance Graphs

The following graphs show the performance of the three algorithms for the following values:

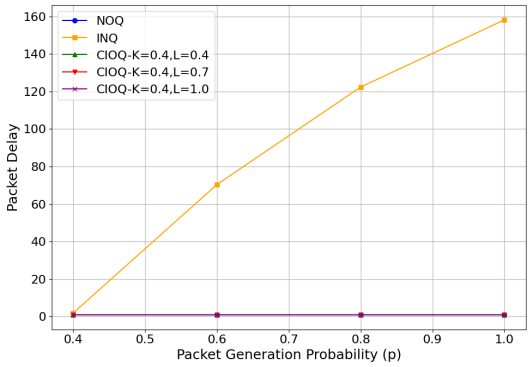
- Number of Ports (N):
  - $N = 32$
  - $N = 64$
- Packet Generation Probability (p):
  - $p \in \{0.4, 0.6, 0.8, 1.0\}$
- Parameter K for CIOQ:
  - $K \in \{0.4, 0.7, 1.0\} \times N$
- Parameter L for CIOQ:
  - $L \in \{0.4, 0.7, 1.0\} \times N$

The above graphs can be interpreted as follows:

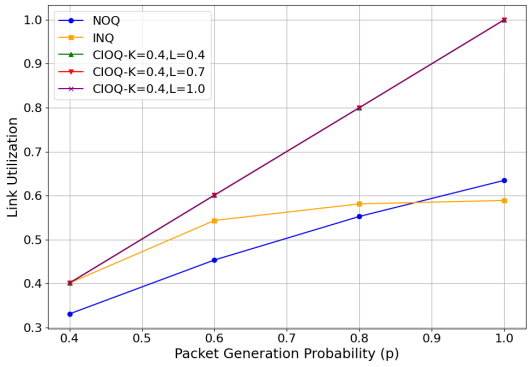
- As depicted in the figure above, the CIOQ link utilisation and packet delay achieves maximum theoretical value of p (that is, along $y = x$ line), and packet delay of 1 for $K = 0.4, L = 0.4$ itself. Therefore, increasing K or L further will not improve the statistics any further. Thus, only a few curves for different values of L is plotted for clarity sake.
- As we can see from the figure above, the utilisation of CIOQ is best. For lower loads (values of p), INQ performs better than NOQ, but for higher values NOQ performs better.



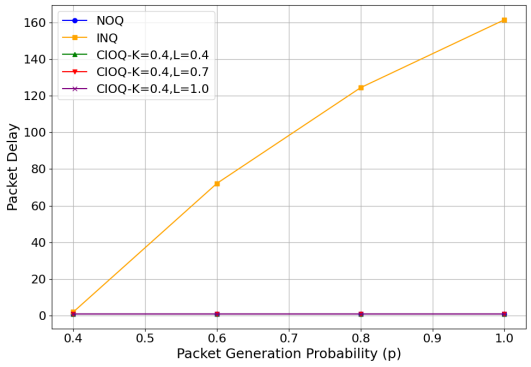
(a) Utilization for $N = 32$



(b) Delay for $N = 32$



(c) Utilization for $N = 64$



(d) Delay for $N = 64$

Figure 2: Performance graphs for different values of N.

- We also see that the delay of CIOQ and NOQ both are 1, but for INQ, the delay increases a lot with higher values of P.
- Another important observation is that, for $N = 32$ and $N = 64$, the graphs are exactly identical. This is probably because we are comparing the averages over a long time period, with $T = 10000$ time slots and a large buffer size of 100.

7 Conclusion

The iSLIP performs much better than No Queue (NOQ) and Input Queueing with FIFO scheduling (INQ) on link utilisation and drop probability metrics. Further it also achieves packet delays of nearly 1 time slot not unlike NOQ. Among INQ and NOQ, INQ suffers from much higher packet delays. At lesser loads, INQ outperforms NOQ in drop probability and link utilisation, but NOQ surpasses INQ at higher loads, as is expected from the theoretical results.

References

[1] N. McKeown. "The iSLIP scheduling algorithm for input-queued switches". In: IEEE/ACM Transactions on Networking 7.2 (1999), pp. 188–201. DOI: 10.1109/90.769767.

[2] M. Karol, M. Hluchyj, and S. Morgan. "Input Versus Output Queueing on a Space-Division Packet Switch". In: IEEE Transactions on Communications 35.12 (1987), pp. 1347–1356. DOI: 10.1109/TCOM.1987.1096719.

[3] Long Gong et al. "SERENADE: A Parallel Iterative Algorithm for Crossbar Scheduling in Input-Queued Switches". In: 2020 IEEE 21st International Conference on High Performance 2020, pp. 1–6. DOI: 10.1109/HPSR48589.2020.9098995.

List of Figures