# A Report on
# "Performance of Queueing in a Packet Switch"

CS544 : Topics in Networks
Indian Institute of Technology Guwahati

Aditya Trivedi & Atharva Vijay Varde
190101005 & 190101018

## 1 Introduction

In this assignment, we have compared various queue scheduling techniques such as INQ, KOUQ and ISLIP. These have been implemented in C++.

Switch operation consists of three phases :

1. Traffic Generation
2. Packet Scheduling
3. Packet Transmission

### 1.1 Traffic Generation

The unit of traffic is a packet. It is defined as a struct :

```
struct packet{
    int idx;
    float st_time;
    float en_time;
    int ip_port;
    int op_port;
};
```

In each time slot, for each input port, a packet is generated with some probability. This is implemented by comparing a randomly generated value (between 0 and 1) to a *packetgenprob*.
Each input and output port has an associated queue represented as a vector of queue of packets.

```
vector<queue<packet>> ip_port;
vector<queue<packet>> op_port;
```

Whenever a packet is generated, its output port is randomly selected, and the packet is placed in the input queue, **if** the input buffer is not exceeded. In case of ISLIP, packet is inserted in corresponding output buffer.

```
// if ISLIP, send it to Virtual Output Queue
if(queue_type == "ISLIP"){
    int csize = ip_port_voq[i][op_port].size();
    if(csize >= buffer_size){
        // input buffer full, drop the packet;
    }else{
        cout<<"Packet from "<<i<<" to "<<op_port<<" generated."<<endl;
        ip_port_voq[i][op_port].push(tmp);
    }
}else{
    int csize = ip_port[i].size();
    if(csize >= buffer_size){
        // input buffer full, drop the packet;
    }else{
        ip_port[i].push(tmp);
    }
}
```

## 1.2 Packet Scheduling

In this phase, packets generated in phase 1 are scheduled for transmission. In this assignment we cover 3 different scheduling techniques, they are as follows :

1. **INQ** : At each output port, if there is only 1 incoming packet, transmit it directly. Else randomly select 1 packet for transmission. Rest of the packets remain in the input buffers.
2. **KOUQ** : At each output port, at most `K` (*knockout*), number of packets are pushed to the output buffer. Remaining are **dropped**.
3. **ISLIP** [McK99] : Each input port sends ***requests*** to each output port, for which it has a packet in the `virtual output queue`. Each output port which receives requests, ***grants*** 1 request based on a priority, decided by *round robin* fashion. Each input which receives grants, ***accepts*** 1 request based on a priority, decided by *round robin* fashion. This accepted packet is transmitted and the round robin priority is updated at both input and output port.

Each of these techniques is implemented as `phase_2_inq()`, `phase_2_kouq()` and `phase_2_islip()` respectively.

## 1.3 Packet Transmission

In this phase, packets at the head of each of the output queue are transmitted. The delay of the packet and utilisation of output queue is noted for simulation statistics.

# 2 Performance Analysis

In this section, performance of all three scheduling schemes are compared. Additionally, the performance is compared when parameters such as `N`(*number of ports*), `B`(*buffer size*), `K`(*knockout*), `p`(*packet generation probability*) are varied.

## 2.1 INQ Scheduling Scheme

- Whenever there is a contention for an output port, one packet is randomly selected for transmission while others wait in their input queues. Since this is a very basic scheduling scheme, the link utilization will not be very high. The following graphs demonstrate the variations of packet delay and link utilization with buffer size and number of ports :

- Low Probability Case: p= 0.5.

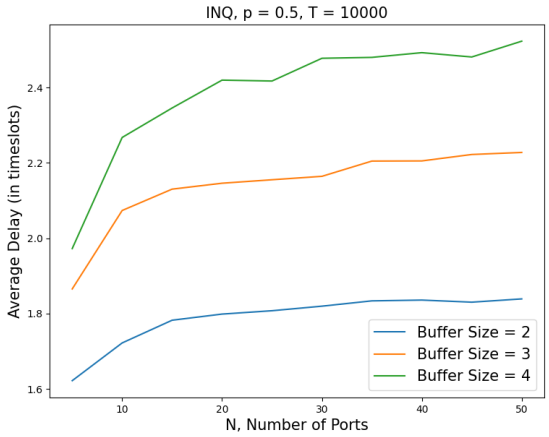


Figure 1: Packet delay

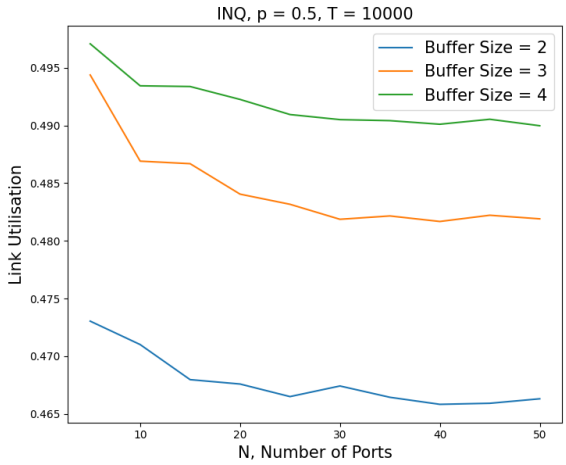


Figure 2: Link Utilization

From the graphs, it is visible that an increase in N corresponds to an **increase in average delay** per packet, as well as **decreased link utilization**. This is because an increase in the number of ports means more chances of collisions at the output ports.
Also, as the size of buffer (B) increases, the packet delay **increases**, since packets which would have been dropped previously remain in the queue now, and get transmitted after some time. However, the **overall link utilization increases** since a greater number of packets in the buffer keeps the output ports busier.

- High Probability Case : p=1. Here, packets are definitely generated at each input port in each slot.

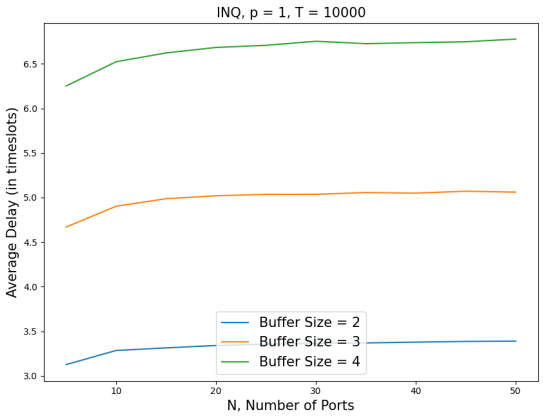


Figure 3: Packet delay

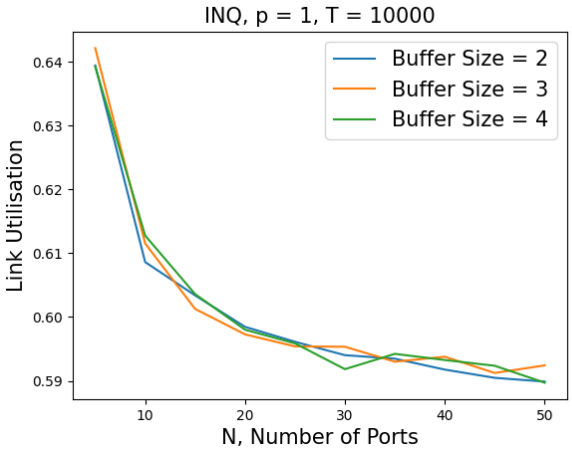


Figure 4: Link Utilization

Similar trends are also observed here. With an increase in N, packet delay increases and link utilization decreases. Packet delay increases with an increase in buffer size.
Note that :

- For p=1, the average delay is significantly higher than for p=0. This is because the input queues are much more occupied when p=1.
- For p=1, the average link utilization is also higher (64% vs 50%). This is also because of higher number of packets generated.
- Link utilization does not depend significantly on buffer size. This is because since such a high number of packets are generated, the buffers are full in a large number of slots. Thus, the size of the buffers becomes irrelevant.

## 2.2 KOUQ Scheduling Scheme

KOUQ uses output queueing to make the packet scheduling more efficient. In this case, the switching fabric has to be a lot faster than the output ports, since during one scheduling phase, multiple packets are sent out from the same input port.
Note : We have taken the total size of output buffer as B, and the parameter K has been used for selecting the packets to be dropped. i.e. If x packets are destined for an output port, K of them are first randomly selected, and then they are added to the buffer UNTIL the buffer is full. The remaining packets are dropped.

Graphs of packet delay and link utilization with changes in N and B are given below, for different values of K and p :

1. K = 0.6, p= 0.5

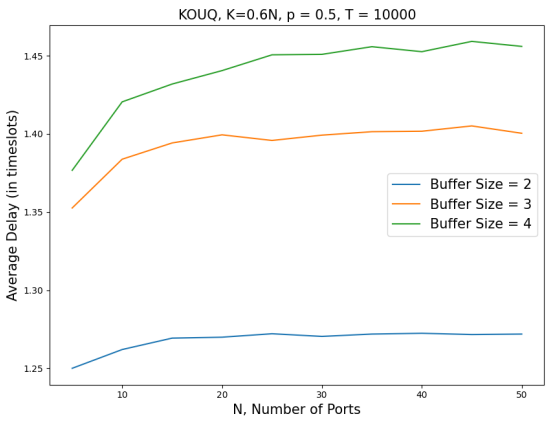


Figure 5: Packet delay

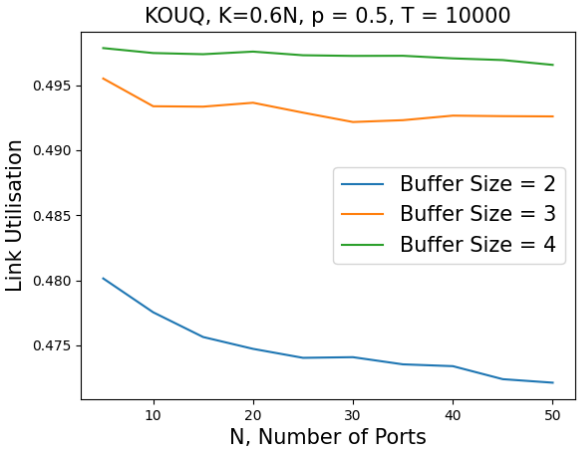


Figure 6: Link Utilization

Again, with an increase in N, **packet delay increases while link utilization decreases**. Also, with an **increase in buffer size, packet delay increases (since lesser packets are dropped) but link utilization increases**.

Already, if we compare INQ scheduling with KOUQ scheduling, we can see that :

- Packet delay in KOUQ is much lesser than in INQ. This is because of two reasons :
  (a) KOUQ dispatches a much larger number of packets to output ports in a single slot
  (b) KOUQ drops many packets instead of storing them indefinitely like in INQ.
- Link utilization in KOUQ is slightly better than INQ at low probability.
- The packet drop probability in KOUQ is non-zero.

2. K=0.6, p=1

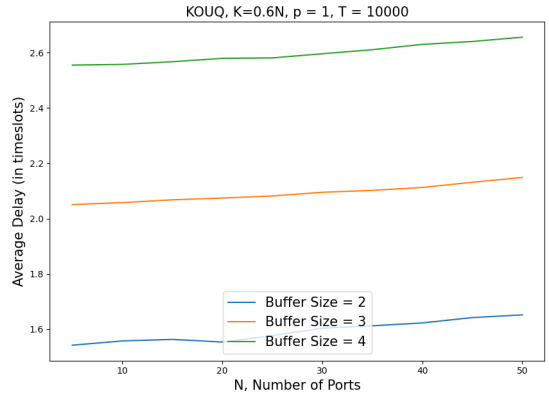


Figure 7: Packet delay

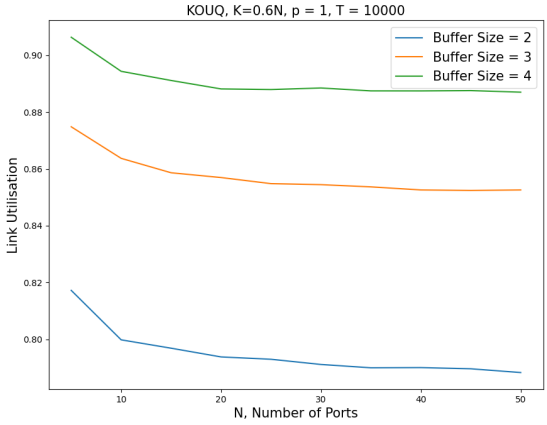


Figure 8: Link Utilization

**With an increase in N, packet delay increases while link utilisation slightly decreases. With an increase in buffer size, packet delay increases significantly, but link utilisation also increases**.

At high probability, we can see that KOUQ clearly outperforms INQ in terms of packet delay and link utilisation (compare with figures 3 & 4). However, at higher probabilities, the percentage of packets dropped will also increase.
We now check the variations in packet delay and link utilisation if we vary K.

3. K=0.8, p = 0.5

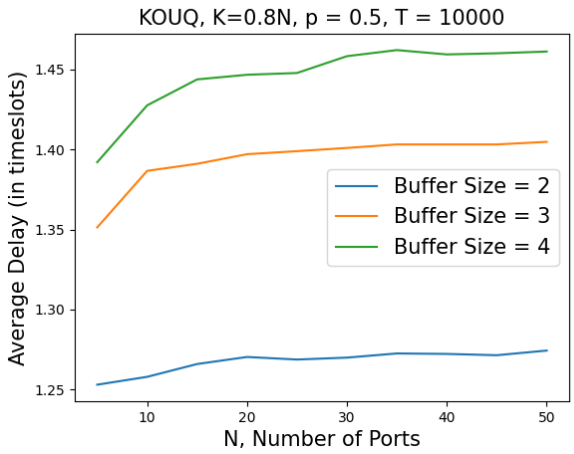


Figure 9: Packet delay

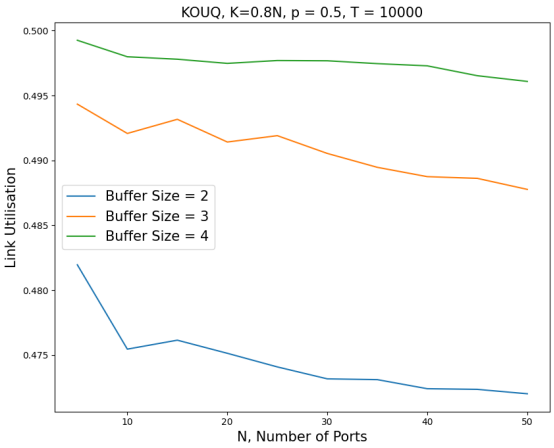


Figure 10: Link Utilization

The following observations can be noted:

- With increase in N, packet delay increases and link utilization decreases noticeably.
- With increase in buffer size, packet delay increases but link utilization also increases.
- **If we compare this case to K=0.6, we notice that with increase in K, link utilization has increased slightly, and packet dropping has decreased.** However, an increase in K causes a load on the hardware (since more buffer space is required).

4. K=0.8, p = 1

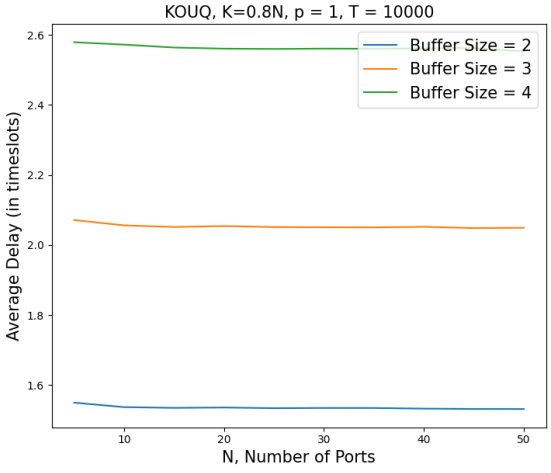


Figure 11: Packet delay

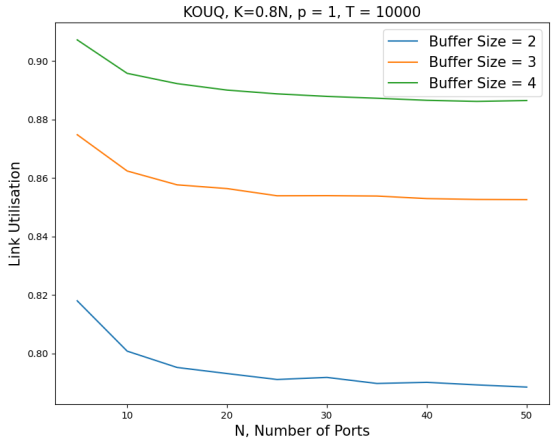


Figure 12: Link Utilization

Observations :

- At high probabilities, packet delay is almost constant with increasing N. **This shows that while N will increase, the excess packets will go on getting dropped and thus will not impact the probability**.
- As B increases, packet delay increases but L.U also increases.

5. K=1, p = 0.5

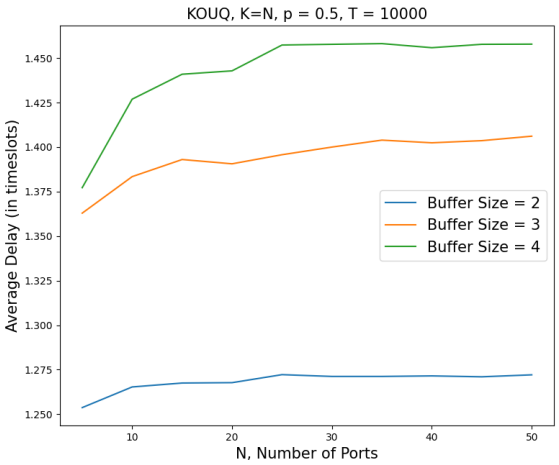


Figure 13: Packet delay

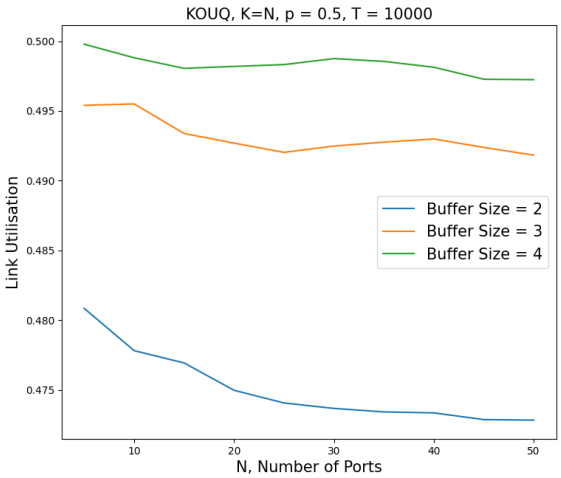


Figure 14: Link Utilization

Observations:

- As N increases, P.D increases and L.U decreases.
- As B increases, P.D increases, L.U increases.
- **Here, a sharper increase in packet delay w.r.t. N is observed due to increased value of K (bigger buffer leads to lesser packets dropped)**.

6. K=1, p = 1. Here the buffer size = N, and the number of packets generated are also large.

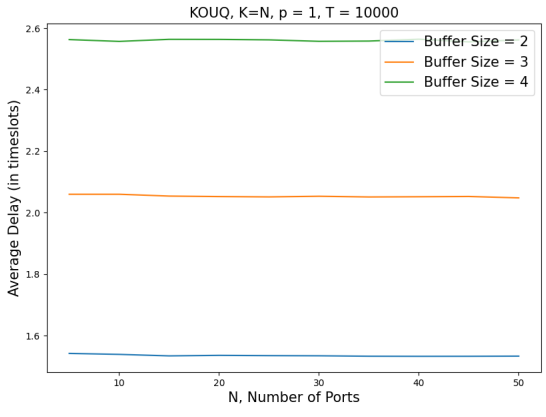


Figure 15: Packet delay

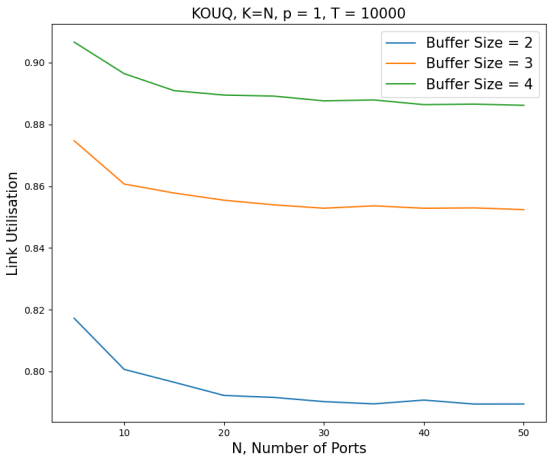


Figure 16: Link Utilization

Observations :

- Again, packet delay remains almost constant with varying N.
- With increasing B, P.D increases and L.U also increases.

**Note :** The values of link utilization in KOUQ are very high in almost all the cases. This is because KOUQ sends packets from a single input port to multiple output ports during a single slot. However, this is not necessarily a measure of great performance since:

1. Packets are dropped more frequently.
2. A sufficiently fast hardware is required which can cope up with the scheduling algorithm.

## 2.3 ISLIP Scheduling Scheme

The ISLIP algorithm attempts to use the concept of virtual output queueing to avoid Head Of Line Blocking. Each input port can be said to consist of N virtual queues, each corresponding to one of the output ports. Thus, packets destined for output port 1 will not be blocked by packets destined for any other output port.
Also, to decide which request the output port will grant, and which grant the input port will accept, we use a round-robin type priority variable.
**Thus, the ISLIP algorithm tries to maximise efficiency without putting too much load on the switching fabric, as well as avoiding packet drops.**

1. Low Probability; p=0.5.

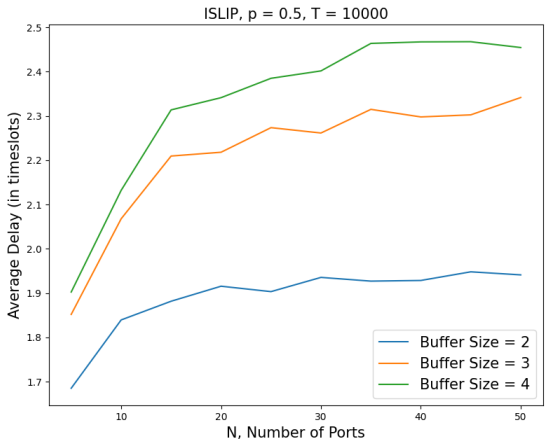


Figure 17: Packet delay

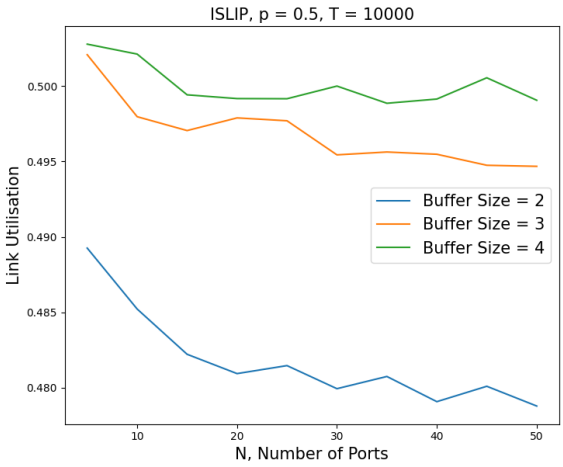


Figure 18: Link Utilization

Observations :

- Again, with an increase in N, packet delay increases and link utilization decreases. However, here the graph fluctuates a bit more than in the above two algorithms. **This is because in ISLIP, the request-grant pattern is not random but based on fixed priorities.** Thus, it will give slightly non-uniform results.
- With an increase in B, P.D increases and L.U decreases.
- **Note that ISLIP is already outperforming the INQ algorithms without any requirements from the hardware. This is the advantage of virtual output queueing.**

2. High Probability; p=1.

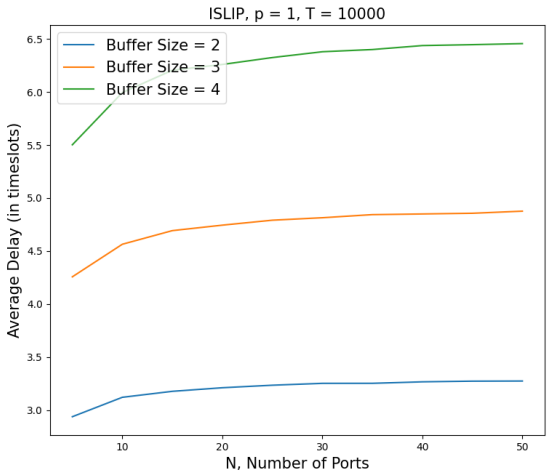


Figure 19: Packet delay

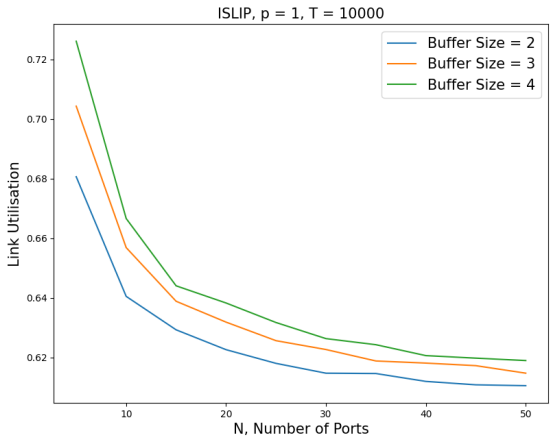


Figure 20: Link Utilization

Observations :

- As N increases, P.D remains almost constant and L.U decreases.
- As B increases, P.D increases but L.U also increases.
- **At high probabilities, the performance of ISLIP begins to match KOUQ. However, the packets dropped in ISLIP are much lesser.**
- The packet delay is remaining almost constant in ISLIP since it is a fair algorithm, giving chance to every port.

Performance of ISLIP with variations in p

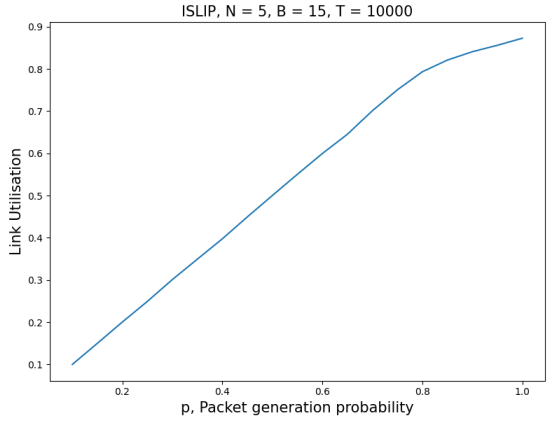


Figure 21: Packet delay

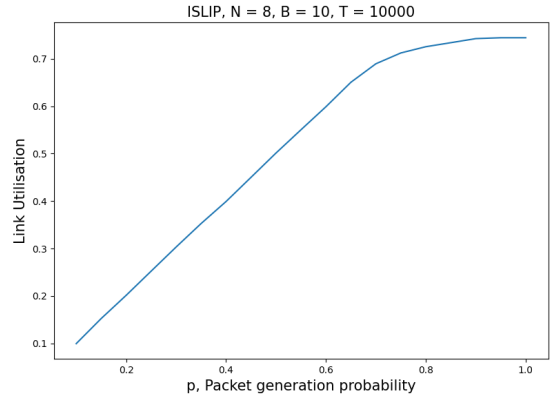


Figure 22: Link Utilization

At higher probabilities, i.e. **bursty traffic**, the ISLIP algorithm performs exceeding well, **reaching 0.9 utilisation**. This is because the virtual output queuing technique aims to find the maximal matching. This leads to maximal amount of packets being transmitted.

# 3 Conclusion

In the INQ scheduling technique, we broadly conclude that link utilisation, decreases with increase in number of ports and increases with increase in buffer size. The packet delay increases with increase in buffer size and increase in number of ports. This can be attributed to the fact that the large the buffer, the more packets will remain waiting **(rather than being *dropped*)**.

KOUQ scheduling technique has similar trends as INQ. However the link utilisation is much better than INQ, since packets are being dropped instead of keeping them waiting at the input buffer queues. During a single time slot, multiple packets are dispatched from a single input port, although this increases the link utilisation, it requires a much faster (**K times faster switch fabric**) to cope up with higher throughput. This incurs *higher hardware cost*. Varying knockout `K` from 0.6 to 1.0, we find that at higher values, fewer packets are dropped but more hardware space is required (*to accommodate larger buffers*).

Finally, in the ISLIP technique, which is simple in terms of software and hardware implementation, and is also fair in w.r.t. input ports (due to ***modified round robin scheduling***). ISLIP outperforms INQ without incurring hardware costs in the switch fabric. This is the advantage of virtual output queues as mentioned above. Here we have implemented single iteration ISLIP which can be improved by running multiple iterations. This will lead to a tradeoff between delay and throughput.

# Assumptions

- The output buffer (in case of KOUQ) can only accommodate up to B packets.
- The total size of input buffer, including all virtual output queues is B (in case of ISLIP).

# Note

All the output files can be found in the `/output` directory and the figures and scripts can be found in `/pics` and `/scripts` respectively. A `readme` file is provided for compilation and execution details.

# References

[McK99] N. McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201, 1999.