Topic in Networks

Assignment on the performance of queuing in a packet switch

SUBMITTED BY
Priyank Soni 190101067
Rohit Kumar 190101077

OBJECTIVE

The objective of this assignment is to understand the performance of queuing in a packet switch by implementing the following scheduling mechanisms: INQ, KOUQ ,iSLIP.

SWITCH OPERATIONS:

The three phases of the program are:

- **Traffic Generation:**

    1. In this phase, each port will generate a packet with probability packetgenprob and an output port for the packet will be selected randomly.
    2. If the packet is generated, it will be pushed to the input queue only if the queue size is less than buffer size.
    3. The structure of the packet is defined by “struct packet” and the input queues are defined by a vector of queues, where each queue in the vector corresponds to one input port.

```
// packet attributes
struct packet
{
    int id;// packet id
    bool is_generated; //if packet is generated or not
    int ip; //input port
    int op; //output port
    double gen_time; //geneartion tym of packet
    double comp_time; //completion time of transmission
    //We will calculate delay by differenece of above two
};

vector<queue<packet>> input_ports;
vector<queue<packet>> output_ports;
```

- **Scheduling**

    1. In this phase, the packets generated in the previous phase are handled.
    2. INQ: If there is no contention for its desired output port, it is selected for transmission and placed in the corresponding output port’s buffer. Else one packet is selected randomly for transmission and the rest are queued at the input port. No packet is dropped. Random selection is done using the rand() function.
    3. KOUQ: This scheduling mechanism can transmit at most K packets in a given time slot for any output port. If more than K packets arrive at any output port, K are selected randomly and the rest are dropped. For selecting K random numbers out of total contending packets, we make an empty set and insert elements randomly into it until the set size becomes K. This way we get K unique random packets.
    4. iSLIP: In mechanism, every input port has a virtual output queue for every output port. The data structure used for this is vector of vector of queues. After this there are three steps involved:

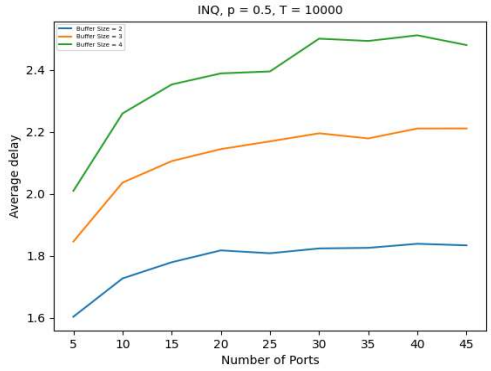
A. Request: Each input port sends a request to every output port for which it has a queued packet.
B. Grant: Each output port selects one of the input ports in a round robin fashion and grants its request. The round robin pointer is incremented only if the granted request is accepted.
C. Accept: Each input port then accepts one of the granted requests in a round robin fashion.

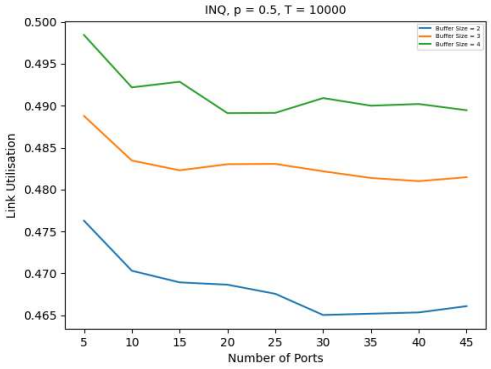
- **Transmission**

1. In the Transmission phase, the actual transmission of the selected packet is done from the head of the queue to an output port.
2. Delay and Link utilization are also calculated accordingly.

Analysis of INQ

1. p=0.5

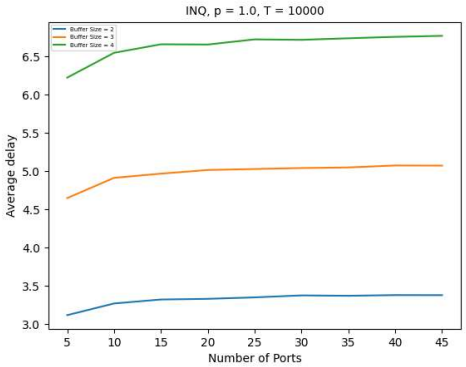


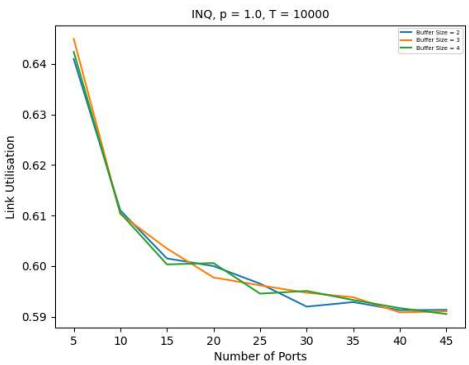


OBSERVATION:

a. average delay increases with increase in number of ports
b. average link utilization decreases with increase in number of ports
c. Both average delay and average link utilization increases with increase in buffer size

2. p=1.0





OBSERVATION:

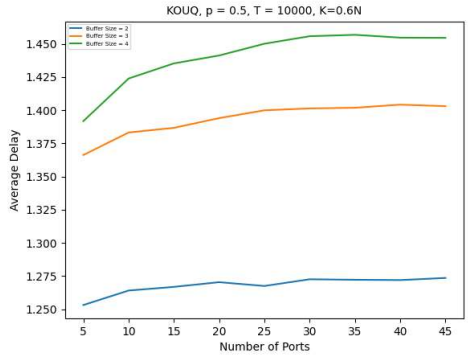
a. average delay increases slightly with increase in number of ports.
b. average link utilization decreases sharply.
c. average delay increases with increase in buffer size.
d. Only a marginal difference in average link utilization with buffer size.

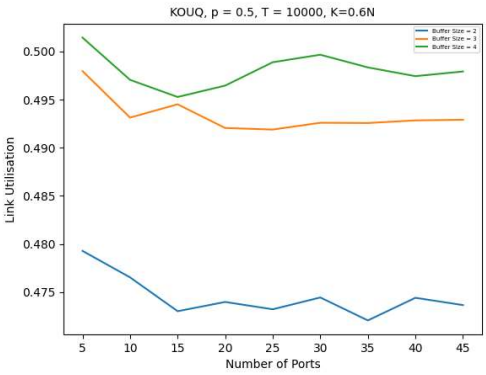
KEY OBSERVATIONS:

1. As the number of ports increases, HOL blocking increases (proven result). Hence the delay also increases. Also, more packets are dropped so less number of packets are transmitted so utilization decreases. This is true in all three scheduling algorithms in this report.
2. As p increases from 0.5 to 1, more packets arrive in the queue so each packet has to wait longer for its transmission. So the delay increases. Also, more packets are transmitted so utilization also increases.

Analysis of KOUQ

1. p=0.5, k=0.6N

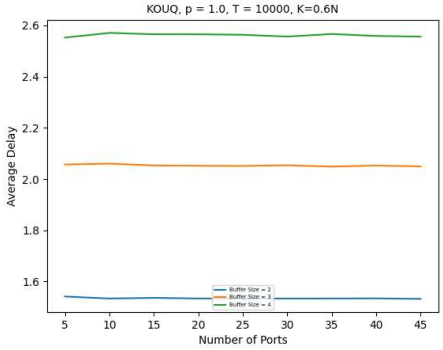


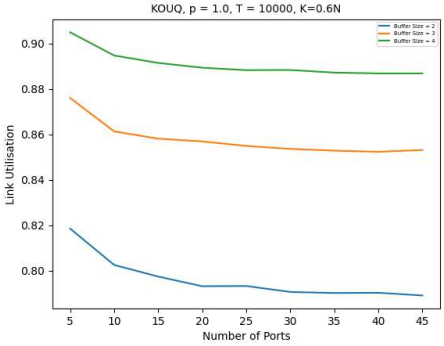


OBSERVATION:

a. average delay increases with increase in number of ports
b. average link utilization decreases with increase in number of ports
c. Both average delay and average link utilization increases with increase in buffer size.

2. p=1, k=0.6N

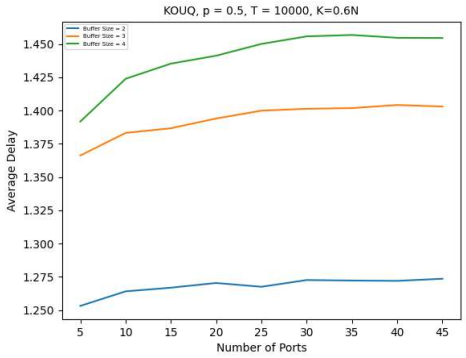


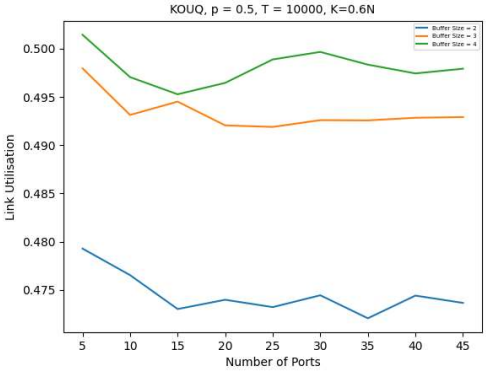


OBSERVATION:

a. average delay almost remains constant with increase in number of ports.
b. average link utilization decreases with increase in number of ports.
c. Both average delay and average link utilization increases when we increase buffer size.

3. p=0.5, k=0.8N

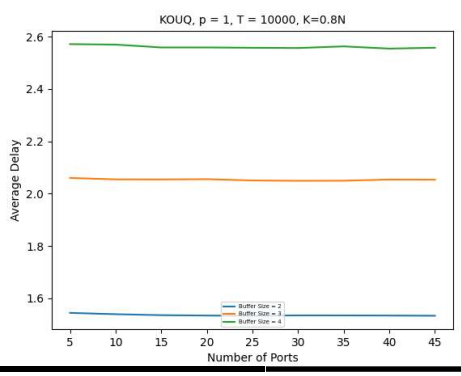


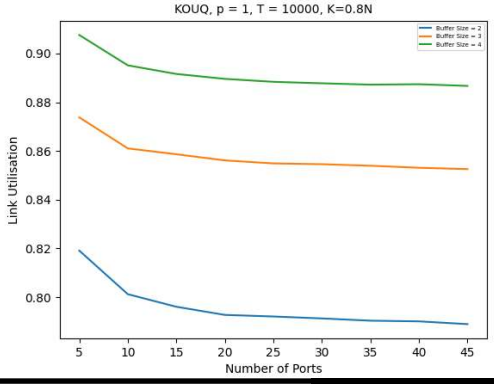


OBSERVATION:

a. average delay increases with increase in number of ports
b. average link utilization decreases with increase in number of ports
c. Both average delay and average link utilization increases with increase in buffer size.

4. p=1.0, k=0.8N

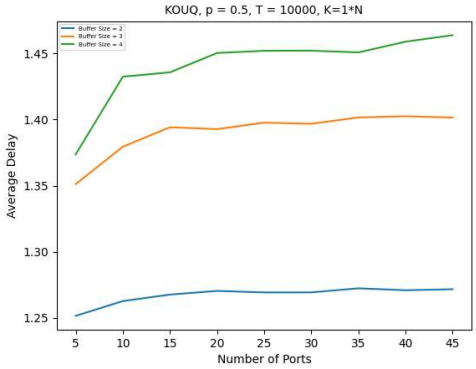


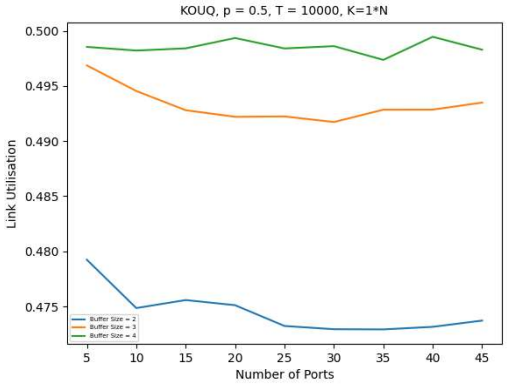


OBSERVATION:

a. average delay almost remains constant with increase in number of ports.
b. average link utilization decreases with increase in number of ports.
c. both average delay and average link utilization increases with increase in buffer size.

5. p=0.5, k=1*N

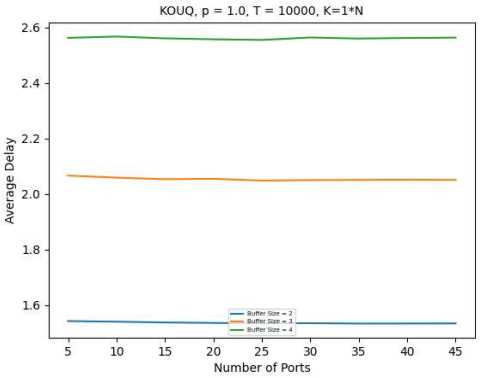


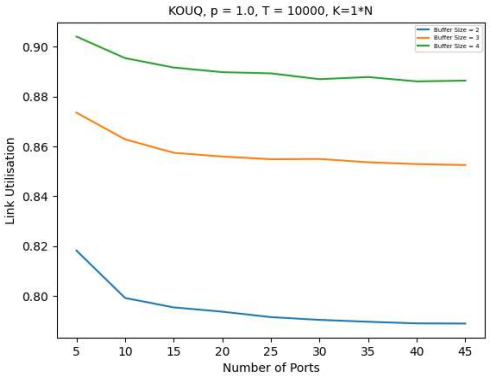


OBSERVATION:

a. average delay increases slowly with increase in number of ports.
b. average link utilization decreases with increase in number of ports.
d. both average delay and average link utilization increases with increase in buffer size.

6. p=1.0, k=0.8N





OBSERVATION:

a. average delay almost constant with increase in number of ports.
b. average link utilization decreases with increase in number of ports.
c. both average delay and average link utilization increases with increase in buffer size.
d. Link utilization proportional to packet generation probability

KEY OBSERVATIONS:

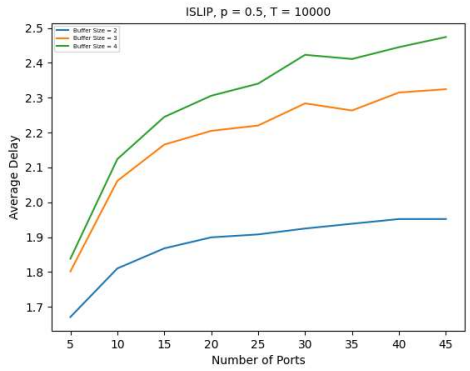
1. Keeping everything **except p constant** and increasing p, we observe that delay and utilization both increase. This is because more packets are arriving while the buffer size is the same, so packets have to wait longer in the queue.

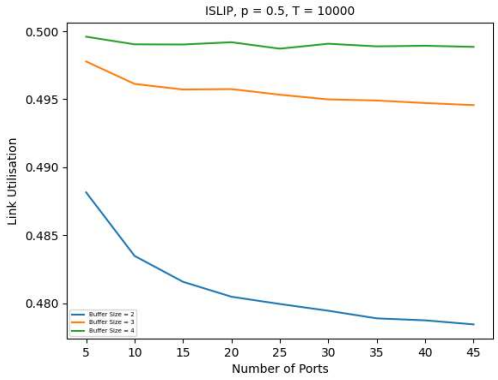
2. Keeping everything **except K constant** and increasing K, we observe that there is no significant difference in delay and utilization.

3. Keeping everything **except buffer size constant** and increasing buffer size, we observe that delay and utilization both increase. This is because less packets are dropped so more packets are there in the queue , so packets have to wait longer in the queue and more packets are transmitted through the link.

Analysis of ISLIP

1.p=0.5

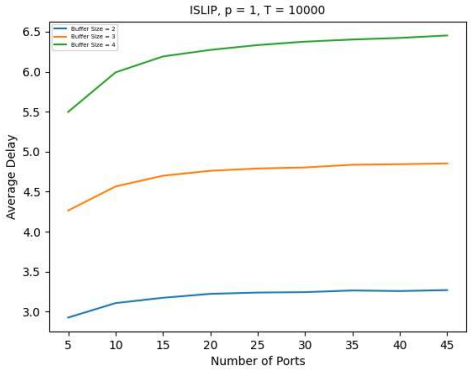


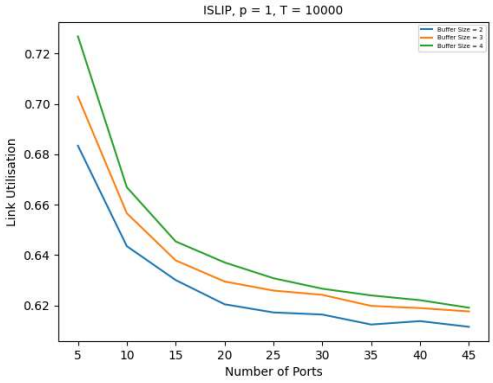


OBSERVATIONS:

a. Average delay increases with number of ports
b. Average Utilization slightly decreases with number of ports
c. Both delay and utilization increase with buffer size.

2. p=1.0





OBSERVATIONS:

a. Average delay slightly increases with number of ports
b. Average utilization drastically decreases with number of ports
C. Both delay and utilization increase with buffer size.

KEY OBSERVATIONS:

1. Keeping everything **except p constant** and increasing p, we observe that delay and utilization both increase. This is because more packets are arriving while the buffer size is the same, so packets have to wait longer in the queue. Also more packets pass through the switch so utilization increases.

2. Keeping everything **except buffer size constant** and increasing buffer size, we observe that delay and utilization both increase. This is because less packets are dropped so more packets are there in the queue , so packets have to wait longer in the queue and more packets are transmitted through the link.

Conclusion

Analyzing the three algorithms, we observe that certain statements are true in all three cases. With increases in the number of ports, delay and utilization both increase. Similar is the case with buffer size(not in INQ) and p.

Comparing these algorithms with each other we get the following conclusions:

- In INQ, delay increases and link utilization decreases with increase in port size. So, INQ is not a good choice if we want high number of ports or large buffer size which is a probable case in real life scenarios. Maximum utilization achieved when p=0.5 of 50% and 64% when p=1.

- In the case of KOUQ, we got better performances than that of INQ, the average delay drops down as compared to INQ as seen in the performance graphs. With the decrease in buffer size there is a rapid decrease in average delay. Even though delay is small but packet drop is significantly high and the architecture is very costly. Such a high rate of packet drop is not desirable in real networks.

- In ISLIP, delay slightly increases with number of ports, but packet drop probability is very less. For the same drop probability, delay is lesser in iSLIP than in KOUQ. Also it gives fair chance to every port using round robin and hence there is no starvation.
  ISLIP also outperforms INQ uniformly. It also performs better than KOUQ(at higher p values). Even though the link utilisation of iSLIP is lesser than that of KOUQ, iSLIP requires less intensive hardware, and also avoids packet dropping. Furthermore, in ISLIP a maximal match is found between input and output ports, increasing the efficiency. So it is the most preferred scheduling mechanism out of these three.