CHAPTER 2

LITERATURE SURVEY

Numerous schedulers have been investigated and implemented by both academia and industry researchers to support high speed internet switch architectures. All proposed distributed and parallel scheduling algorithms for Input Queued Switches and Buffered Crossbar Switches operated under unicast and multicast environments are studied and analysed in this chapter. A commonly known simplest scheduling scheme is First in First out (FIFO) scheduler proposed for IQ switches and BCS [32]. It uses a single FIFO queue in each port of the BCS. At each timeslot, a HOL cell from each FIFO queue is selected for scheduling. Moreover as per the given requirement the output port can accept only one cell in a given time period. The problem is, if two cells are destined for the same output port then only one cell is selected and the other results in HOL blocking. This reduces the performance of the switch considerably and cannot be used with high speed switches.

2.1 SCHEDULING ALGORITHMS: A STUDY

In this section, a study is made on various crossbar switch scheduling algorithms.

2.1.1 Maximum Weight Matching Algorithms

In Maximum Weight Matching (MWM) algorithms, a weight is attached with each request from inputs to outputs. Selection of cell to be scheduled is based on the maximum weight. Some of the commonly used MWM algorithms are Longest Queue First, Oldest Cell First and Longest Port First algorithms.

*Longest Queue First Algorithm* - Longest Queue First (LQF) algorithm is a commonly used input scheduler proposed by [33]. It takes the number of cells in each
VOQ as weight. The algorithm selects an eligible cell from the VOQ which has the highest weight and places it in the crosspoint queue. Simulation result shows LQF-LQF offers better throughput and delay performance [34] compared to other MWM scheduling algorithms. But it suffers from starvation as the algorithm does not consider the waiting time of cells in the VOQ and therefore cells in the small sized queue needs to wait for prolonged period of time. Time complexity of LQF algorithm is given as $O(\log n)$.

**Oldest Cell First Algorithm** - In an attempt to reduce starvation further, Oldest Cell First (OCF) algorithm was proposed by [35]. OCF uses the waiting time of HOL cells as weights. The OCF algorithm selects a match such that the sum of all served queues waiting time is maximized. Unlike the LQF algorithm, the OCF algorithm does not starve any queue and unserved HOL cells will eventually become old enough to be served. However the throughput and delay performance of OCF is very less comparing to LQF. So a better algorithm is required to avoid the starvation and provide maximum throughput and delay performance.

**Longest Port First Algorithm** - Another well known MWM algorithm is Longest Port First (LPF) algorithm proposed by [36]. Because of high computation complexity in the order of $O(N^3 \log n)$, both LQF and OCF are not feasible for hardware implementation. LPF offers better computation complexity and can be implemented in hardware with high speed. Total number of cells queued in the input and output interfaces through which a cell travels constitute a weight. HOL cell with maximum weight is selected for the schedule. Simulation results show that LPF can achieve high throughput [37] under any admissible traffic compared to the other MWM scheduling algorithms. Small number of overflows occurs when the arriving
traffic is non-uniform and is a drawback in LPF. LPF has a runtime complexity of $O(N^{2.5})$ which is lower than LQF.

MWM scheduling algorithms achieve 100% throughput under any admissible traffic. However, their good performance and stability comes at the expense of high computation complexity.

### 2.1.2 Approximating Maximum Size Matching Algorithms

Approximating Maximum Size Matching algorithms are iterative and therefore simple to implement as hardware. AMSM works faster to cope up with today’s technologies. Some of the AMSM scheduling algorithms are:

**Parallel Iterative Matching Algorithm** - The Parallel Iterative Matching (PIM) algorithm [15] attempts to approximate a maximum size matching algorithm by iteratively matching the inputs to the outputs until it finds a maximum size match. Each iteration consists of three steps:

**Request** - Each unmatched input sends a request to every output for which it has a queued cell.

**Grant** - If an unmatched output receives any requests, it grants one by randomly selecting from all requests.

**Accept** - If an input receives more than one grant, it accepts one by random selection.

Drawbacks in PIM are:

- Randomness is difficult and expensive to implement at high speed. Each arbiter must make a random selection among the members of a time-varying set.
• When the switch is oversubscribed, PIM can lead to unfairness between connections.

• PIM does not perform well for a single iteration. It limits the throughput to approximately 63% which is slightly higher than a FIFO switch.

**Round Robin Matching Algorithm** - RRM is the simplest algorithm proposed by [32] for input queued switches. RRM work in three steps: Request, Grant and Accept.

**Request** - Each input sends a request to every output for which it has a queued cell.

**Grant** - If an output receives any requests, it chooses the one that appears next in a fixed Round-robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest priority element of the round-robin schedule is incremented to one location beyond the granted input.

**Accept** - If an input receives a grant, it accepts the one that appears next in a fixed round-robin schedule starting from the highest priority element.

RRM delivers average delay performance under uniformly independent and identically distributed traffic. Updating the pointers at output arbiters is one of the reasons for poor performance. Synchronisation of the grant pointers also limits performance with random arrival patterns

**iSLIP Algorithm** - Nick Mckeown proposes iSLIP [38] which is an iterative, round robin (RR) scheduling algorithm for input queued switches. iSLIP improves upon Round Robin Matching (RRM) by reducing the synchronisation of the output arbiters. The Grant step of RRM is changed for iSLIP as follows: “If an output receives any requests, it chooses the one that appears next in a fixed round-robin schedule, starting from the highest priority element. The output notifies each input
whether or not its request was granted. The pointer to the highest priority element of the RR schedule is incremented to one location beyond input if and only if the grant is accepted”. This leads to the following:

- Lowest priority is given to the most recently made connection
- No connection is starved
- Under heavy load, all queues with a common output have the same throughput.

From the simulation results it is found that iSLIP works similar to RRM and FIFO when the load is less. If the load is high, iSLIP gives better performance compared to RRM and FIFO. The biggest drawback with iSLIP is its complexity in implementing the hardware.

Most of the AMSM scheduling algorithms provide 100% throughput under uniform traffic and fairly good delay performance as well. However, they are not stable under non-uniform traffic.

**Iterative Least Recently Used Algorithm** - Another successful algorithm for IQ switches is Iterative Least Recently Used [32]. It gives highest priority to the least recently used cell and lowest priority to the most recently used. It works exactly oppose to the iSLIP algorithm in terms of ordering the cells in the scheduler list. Once the grant is successful, it selects a cell from the scheduler list which is least recently used.

2.1.3 Longest Queue First-Round Robin Scheduler

Longest Queue First-Round Robin Scheduling (LQF-RR) scheduling approach [33] is proposed as a high throughput scheduling algorithm for BCS with one buffer
per crosspoint. LQF and Round Robin (RR) schedulers are used as input and output scheduling algorithms respectively. RR is selected for output schedule because of its support for high speed fabric [39]. LQF-RR uses fluid model techniques which is already successful for IQ switches. Simulation results show that this approach is stable for the input/output load \( \leq 1/N \). It delivers 100% throughput for much of the admissible loads. When maximum load is supplied through the VOQ at regular intervals the switch produces more delay and looks unstable.

### 2.1.4 Critical Buffer First Algorithm

Authors in [40] proposed Critical Buffer First (CBF) algorithm which uses Youngest internal Buffer First (YBF) algorithm at the input side and Oldest internal Buffer First (OBF) algorithm at the output side of the switch. It is understood that the information on the internal buffers is sufficient for the schedulers to make effective decisions. The functionality of CBF scheme is very much similar to the buffer-less Oldest Port First (OPF) algorithm. The output scheduler ‘OBF’ performs its arbitration based upon the input waiting time function used by OPF. It is defined as the sum of all HOL cells at the input and waiting time for the same output. However, input scheduler YBF uses an opposite arbitration criterion to the output waiting time function used by OPF. Instead of giving priority to the request with the greatest output waiting function, YBF gives priority to the request with the smallest output waiting time instead. To achieve fairness, both YBF and OBF maintain a highest priority pointer to break ties in the presence of conflicts. Each input pointer is initialized to the index of the input (respectively output) it belongs to.

Authors [40] further updated the CBF algorithm as Current Arrival First-Priority Removal (CAF_PRMV) where the input scheduling gives priority to the
newly arriving packets while the output scheduler chooses the recently arrived packets in the internal buffer. CAF_PRMV avoids using state information or weight function in scheduling decisions. It has the capability to overcome the lack of performance under the non-uniform traffic suffered by CBF. At each time slot, Current Arrival First (CAF) algorithm checks whether there is a new cell arriving at the input port. To accomplish this task in the output scheduling, CAF assigns a priority level to each cell leaving the input port. This level will decide the priority of that cell in the output scheduling phase. The output scheduling phase will be much simpler and faster than sorting. Moreover the adoption of priority levels makes the implementation easy and the hardware requirement simple. Simulation result shows that the algorithm can offer maximum throughput performance under any traffic patterns. CAF_PRMV is totally stateless and requires simple hardware while working at very high speed.

2.1.5 Most Critical Buffer First Algorithm

Mhamdi and Hamdi [41] proposed Most Critical Buffer First (MCBF) a high performance scheduling algorithm for Buffered Crossbar Switches. MCBF objective is to reduce the arbitration time comparatively to CBF and CAF_PRMV algorithms. MCBF uses Shortest internal Buffer First (SBF) at the input side and Longest internal Buffer First (LBF) on the output side of the switch. For scheduling, MCBF does not use any input state information, such as VOQs occupancies or VOQs HOL cells waiting time and therefore the scheme is almost stateless. Information used relies only on the internal buffers which is sufficient for the schedulers to make effective decisions and therefore its hardware is simple to implement.
MCBF’s scheduling decision is based on the number of cells in the internal buffers and therefore it takes less time for arbitration. MCBF is tested under bursty uniform and non-uniform traffic models. Simulation result shows that MCBF provides shortest delay performance compared to LQR-RR and OCF-OCF algorithms. Even after 90% of the load is supplied to the switch, MCBF looks very stable. Under non-uniform traffic, MCBF performance can be further improved by increasing the size of the crosspoint buffer.

2.1.6 Randomized Algorithms

In [42], Petar proposed a Randomized Algorithm (RA) for Buffered Crossbar Switches. The motivation for proposing the algorithm is to overcome the complexity posses by Maximum Weight Matching algorithms and to impose stability under any admissible traffic. RA holds unit mean exponential random variables on the input side. By adjusting the intensity of the variables, the input is selected. At some fixed time, for each input-output pair \((i, j)\) a timer is set that expires in \(V_{i,j}(t)\) time units. These units do not correspond to time slots needed to switch a packet from an input to an output. Upon the expiration of the timer \((i, j)\), input \(i\) is matched to output \(j\) unless either of them is already matched. This process continues until all timers expire. The timers are set based on local parameters only. However, the complexity of the implementation is not negligible due to the required sources of randomness. There exist at least two possible ways to obtain randomness:

1. External – a physical device (e.g., photon-based) auxiliary to the switching fabric can serve as a source of randomness (Poisson process); since these devices do not have to be implemented in silicon, they can potentially operate at very high speeds.
2. Internal – randomness can be extracted from packet payloads. The algorithm (Mekkittikul and Mckeown, 1998) with an appropriately chosen weight function has the following desired characteristics: High throughput, Starvation free and Simple to implement. Based on the analysis report, it is found that the algorithm remains stable under high traffic loads and performs competitively in comparison with the MWM algorithm.

2.1.7 Localized Independent Packet Scheduling

Deng Pan and Yuanyuan Yang [43] proposed Localized Independent Packet Scheduling (LIPS) algorithm for BCS. It collects the state information from the localized buffers and uses it for both input and output scheduling decisions. For input schedule, based on the available free buffers, HOL cell is selected from VOQ’s using Fixed Priority or Random Priority or in Round-Robin fashion. Similar procedure is followed for output scheduling too. Simulation result shows that LIPS can achieve 100% throughput for any admissible traffic with a speedup of 2. It also requires buffer space greater than one to achieve the desired output. Therefore the hardware implementation cost of LIPS is quite high to meet the needed requirements.

2.1.8 Distributed Queue Input-Output Scheduler

DIStributed QUEue input-Output scheduler (DISQUO) is a distributed scheduling algorithm (DISQUO) [44] for BCS. It achieves 100% throughput under any admissible arrival traffic with only a one-packet buffer at each crosspoint. In DISQUO, the input actions are performed at the beginning of each time slot and outputs transmit packets from the crosspoint buffers before the end of each time. The output ports have to learn the inputs decisions. The key point of DISQUO is that by observing crosspoint buffers, an input and an output can learn each other’s decisions
implicitly. Simulation result shows that it can provide very good delay performance under different traffic arrivals. The simulation results also show that, by using DISQUO, packet delay is very weakly dependent on the switch size, which means that DISQUO can scale with the number of switch ports. The issue with DISQUO is its high implementation complexity and also it needs to keep state information.

2.1.9 Fair Asynchronous Segment Scheduling

Karimi [45] proposed Fair Asynchronous Segment Scheduling (FASS) which is designed for BCS and works without speedup. Its objective is to achieve constant performance guarantees with reduced crosspoint buffers. The packets are segmented and are transmitted, so as to reduce the crosspoint size. FASS uses independent input and output scheduling scheme and both requires the status of the crosspoint to send or receive segments. For input scheduling, a timestamp is used to note down the input start and end time. A segment is eligible for input scheduling if its virtual input start time is smaller than or equal to the current system time. Similar approach is followed in the output scheduling as well. From the simulation results it is understood that the FASS has strong switch stability by showing that the length of input virtual queues are finite. FASS requires no speedup for the crossbar, reducing the implementation cost and improving switching capacity. Experimental result shows FASS can attain 100% throughput under any admissible traffic but its performance decreases when the segment size increases. This is because with the same simulation time, larger segments sizes have smaller probabilities to finish the transmission of the last segment.
2.1.10 SQUID

Authors in [46] propose Stable Queue Input-output Scheduler with Hamiltonian walk (SQUISH) for crosspoint buffered switches. Each input chooses the crosspoint to send a cell based on buffer occupancy information. Each output simply chooses the longest available queue for service. Then, a comparison with a Hamiltonian walk schedule is made to choose the schedule with a greater weight. SQUISH algorithm achieves 100% throughput for any admissible traffic. The complexity of SQUISH is only $O(\log N)$ which is low enough to make practical implementations feasible. To further reduce the complexity, SQUID is proposed. With the availability of SQUISH and SQUID, the scheduler can serve additional cells in a time slot as compared to a bufferless crossbar because of the loose coupling between input and output scheduling facilitated by the availability of crosspoint buffers. This reduces buffer occupancies, leading to significant delay performance improvement. SQUID can achieve 100% throughput for any admissible traffic, with the minimum required crosspoint buffer size being as small as a single cell buffer. Simulation shows the delay performance of SQUID is close to that of an ideal output-queued switch. Only under highly loaded log-diagonal and diagonal traffic, the delay is greater than an output-queued switch.

2.1.11 Counting Method

Crossbar switches works in two phases such as input and output scheduling. The input scheduling policy gives preference to cells based on the input priority list. Similarly, the output scheduling policy gives preference to cells based on the output priority list. Therefore the cell which has high priority in both the scheduling policy will be switched first. Other cells will result in slackness. To overcome this issue,
Counting Method (CM) is proposed by Pan, et.al [47], uses Group By Virtual Output Queue insertion policy:

i. When a cell arrives to a non-empty VOQ, the cell is inserted in the input priority list just behind the last cell belonging to the same VOQ. This ensures that cells destined to the same output are ordered based on departure order.

ii. When a cell arrives to an empty VOQ, the cell is inserted at the head of the input priority list. The Counting Method using Group By Virtual Output Queue insertion policy provides rate and delay guarantees between each input/output pair. This method suffers when the input traffic is non-uniform and in most of the situations it fails to achieve 100% throughput [48]. Further research is required to sort out these issues.

### 2.1.12 Independent Scheduling Algorithm

In [49], authors propose Independent Scheduling Algorithm (ISA) which is a simple packet scheduling algorithm for buffered crossbar switches. Input scheduling and output scheduling of ISA are independent and distributed in nature. Both the input and the output ports make scheduling decisions based on the state information of the crosspoint buffers. Hamiltonian walk is used on the input side to select one of its VOQ’s which has the highest weight and whose corresponding crosspoint buffer is empty. The weight is calculated by the product of Hamiltonian weight and the queue length of the VOQ. The VOQ with the highest weight is send to the crosspoint buffer. The crosspoint buffer is selected in Round Robin fashion. In output scheduling, when the transmission channel of an output port from the crosspoint buffers is idle then the output port selects a crosspoint buffered packet and saves it in its output queue. Analysis shows that the time complexity in ISA is less because of its independency in
scheduling at input and output ports. ISA provides 100% throughput under uniform traffic patterns. However its performance gets reduced under non-uniform traffic patterns.

### 2.1.13 TATRA Algorithm

Prabhakar and Mckeown proposed TATRA algorithm [50] for multicast switches. TATRA is based on the Tetris model, a popular block-packing game. Input cells are mapped into the Tetris blocks and each input cell is composed with set of output cells. At each timeslot, input cells at HOL are dropped into an empty box which has N slots, one for each output. TATRA transfers one cell from each queue alternatively which includes residue cells as much as possible. It treats all the inputs uniformly whether it is uniformly loaded or non-uniformly loaded or loaded with high priority. Therefore the algorithm provides better fairness than other policies as it always concentrates on the residues.

### 2.1.14 Multicast Cross-points Round Robin Scheduler

In [51], Mhamdi and Hamdi proposed a simple scheduling algorithm Multicast Cross-points Round Robin (MXRR) for Internally Buffered Crossbar (IBC) switches. At input scheduling, HOL cells from each input queue are multicast to the crosspoint buffer. Upon non-availability of buffer, cells are switched in the next scheduling phase. At output scheduling, the first non-empty buffer is selected and its cell are transferred to the output queue. Simulation result shows MXRR has a smaller average delay compared to other algorithms. Performance of MXRR can be further improved when the size of the crosspoint buffer is doubled.
2.1.15 Series of Multicasting Algorithms

Authors in [52] proposed a series of algorithms for BCS which operate under multicast environment. The proposed scheduler was simulated in a buffered crossbar with multiple input FIFO queues. Authors proposed four cell assignment schemes for switch scheduling and are Cell oriented Round Robin Assignment (CRRA), Burst oriented Round Robin Assignment (BRRA), Cell oriented Shortest Queue First (CSQF) and Burst Shortest Queue First (BSQF). At each timeslot, input scheduler selects a cell from input queue to crosspoint buffer based on fanout set. Four schemes such as Round Robin, Minimum Residue First, Maximum Service First and Maximum Ratio of Service First are used as input schedulers depending on requirements. For output scheduling, Round Robin (RR) and Longest Queue First (LQF) scheduling schemes are available. If the size of the cross point buffer is one, then Round Robin scheduler is used, otherwise LQF is used. A 2x8 and 8x8 BCS is used for simulation with i.i.d. and bursty traffic arrivals. Results of different scheduling schemes under 2x8 switch shows RR achieve poorest performance and MSRF achieve the maximum performance. For an 8x8 switch, all schedulers achieve similar performance. For output scheduling, simulation result shows both RR and LQF offer similar performances for any admissible traffics.

2.1.16 Enhanced First In First Out based Round Robin Scheduler

In [53], author proposed a Enhanced first in first out based round robin scheduling algorithm for M x N input queued switches operated under multicast environment. For each input cell, a traffic matrix is generated from fanout information. Multicast scheduling decisions are taken based on the resultant matrix values. A complement matrix is generated in order to find the cells that can be sent to the idle outputs and thereby HOL blocking is reduced. Simulation result shows that
the proposed algorithm can achieve a constant throughput performance under any admissible traffic patterns. Moreover the algorithm ensures very low cell latency too.

2.1.17 Unicast and Multicast Dual Round-Robin Integrated Algorithm

Authors in [54] proposed a Unicast and Multicast Dual Round-Robin Integrated Algorithm (UMDRR) to support both unicast and multicast traffic in input queued switches. Proposed scheduler uses only two phase i.e request and grant scheduling steps. At each timeslot, input send request to output to serve cells either in unicast or multicast manner. Upon buffer availability, output grants permission to input for transferring its HOL cell. UMDRR uses three pointers namely unicast, multicast primary and secondary pointer which are used to provide fairness and reduce HOL blocking. Simulation result shows that the algorithm offers better throughput and latency performance for any admissible traffic.

2.2 COMPARATIVE STUDY ON BCS SCHEDULERS

In this section, Buffered Crossbar Switch scheduling algorithms such as are DISQUO, FASS, MCBF, LQF-RR and ISA are analysed and compared.

2.2.1 Simulation Environment

Simulator JNetworkSim [55] is used to model the buffered crossbar switch of size $4 \times 4$ with a buffer size of 1. No speedup is introduced at any stage of the simulation process. BCS scheduling algorithms such as DISQUO, LQF-RR, MCBF, ISA and FASS are selected for comparative analysis based on literature study. Input for the simulator is supplied through non-uniform i.i.d. traffic and non-uniform bursty traffic models. Almost all the algorithms delivered 100% throughput under uniform traffic models and therefore it is not used for analysis. Performance metrics such as throughput and average cell latency are computed and are analysed [34].
2.2.2. Simulation Results

Throughput of all the algorithms with load=100000 bits per schedule for non-uniform i.i.d. traffic is depicted in Fig.2.1. Result shows LQF-RR scheduler provides maximum throughput comparing to other algorithms by more than 3%. As the load exceeds beyond 50%, throughput of all the algorithms starts dropping. When 100% of load is offered, both LQF-RR and DISQUO are able to attain a maximum of 70% throughput. Fig.2.2 shows the average cell latency of all the algorithms with respect to load. When minimum load (<50%) is offered, LQF-RR cell latency is 10% better than other schedulers. As the load exceeds beyond 50%, all the schedulers’ cell latency are seriously disturbed. Comparing all the algorithms, minimum delay is offered by MCBF which is 30%. FASS delivers a worst delay of <35% when maximum load is offered. There is no much difference between the DISQUO, ISA and LQF-RR. Comparing all the algorithms, LQF-RR provides a better throughput performance and MCBF provides better delay performance throughout the simulation executed under non-uniform i.i.d traffic.

Fig.2.3 shows the throughput of all the algorithms with load=100000 bits per schedule for non-uniform bursty traffic. Result shows LQF-RR out performs all other algorithms by more than 10% and as expected, when load exceeds 50% then the throughput starts dropping. In the worst case, FASS attains 53% of throughput when maximum load is offered. LQF-RR shows consistent throughput performance until 70% load is offered and the performance dropped beyond that load.
Fig. 2.1 Throughput as a function with respect to load for non-uniform i.i.d. traffic

Fig. 2.2 Throughput as a function with respect to load for non-uniform bursty traffic
Fig. 2.3 Average Cell Latency as a function with respect to load for non-uniform i.i.d. traffic

Fig. 2.4 Average Cell Latency as a function with respect to load for non-uniform bursty traffic
Fig. 2.4 shows the average cell latency of all the algorithms under non-uniform bursty traffic model. When maximum load is offered, there is no difference between MCBF, LQF-RR and ISA schedulers where each poses a delay between 5% and 30% whereas FASS delivers a worst cell delay of 48%. Table 2.1 shows the throughput and average cell latency performance of various BCS scheduling algorithms for different load structures.

Table 2.1 Performance of various scheduling algorithms

<table>
<thead>
<tr>
<th>Scheduling Algorithm</th>
<th>Traffic Model</th>
<th>Metric</th>
<th>Load offered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Throughput (%)</td>
<td>25%</td>
</tr>
<tr>
<td>LQF-RR</td>
<td>Non-uniform i.i.d.</td>
<td></td>
<td>82.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Non-uniform busy</td>
<td>Throughput (%)</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>8</td>
</tr>
<tr>
<td>MCBF</td>
<td>Non-uniform i.i.d.</td>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Non-uniform busy</td>
<td>Throughput (%)</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>10</td>
</tr>
<tr>
<td>ISA</td>
<td>Non-uniform i.i.d.</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Non-uniform busy</td>
<td>Throughput (%)</td>
<td>72.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>15</td>
</tr>
<tr>
<td>DISQUO</td>
<td>Non-uniform i.i.d.</td>
<td></td>
<td>73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Non-uniform busy</td>
<td>Throughput (%)</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>18</td>
</tr>
<tr>
<td>FASS</td>
<td>Non-uniform i.i.d.</td>
<td></td>
<td>67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Non-uniform busy</td>
<td>Throughput (%)</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>20</td>
</tr>
</tbody>
</table>
2.3 PROBLEMS IDENTIFIED

i. Most of the BCS scheduling algorithms offers 100% throughput performance with minimum delay under uniform traffic but provides poor performance under non-uniform traffic.

ii. BCS schedulers which failed to control the starvation effect results in poor switch performance

iii. Most of the schedulers required speedup ≥ 2 to improve the performance which results in high implementation cost

iv. Most of the algorithms suffered from HOL cell blocking problem

v. Schedulers suffer from high computation complexity which is impractical to implement in hardware

vi. Certain algorithms look unstable under bursty traffics

vii. Some of the schedulers need state information for scheduling which increases the complexity of the scheme

viii. Randomness is difficult and expensive to implement at high speed

ix. Algorithms lack fairness during scheduling

x. Algorithms offer better performance under unicast traffic but failed to attain the same under multicast traffic

xi. Lack of support to fanout scheme in multicasting scheduling

xii. Lack of scalability to ensure performance guarantees

2.4 OBJECTIVES OF THE DISSERTATION

The objective of this dissertation is to

i. Study the importance of switching architectures and its scheduling algorithms

ii. Do literature survey on buffered crossbar switch scheduling algorithms and analyse its performance
iii. Identify the problems and pitfalls in existing schedulers

iv. Propose new scheduling algorithms for buffered crossbar switches

v. Analyse the throughput and delay performance of the proposed scheduling algorithms under uniform and non-uniform traffic patterns

vi. Analyse the impact of various crosspoint buffer sizes on the proposed scheduler

vii. Study multicasting problem on buffered crossbar switches

viii. Analyse the throughput and delay performance of the proposed scheduler under multicast traffic