CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

This chapter provides a survey of the existing literature on scheduling algorithms for packet switches. The focus is on the literature on the IQ architecture, the CICQ architecture and the Parallel Packet Switches in Section 2.2, followed by a brief discussion of the important switch scheduling algorithms proposed in the literature and implemented in practical routers, in Section 2.3. Section 2.4 presents the conclusions from the literature survey.

2.2 Overview of the Literature

2.2.1 Pre-VOQ Switching

The well-known 58.6% throughput limit in IQ switches that occurs due to the Head-of-Line (HOL) blocking phenomenon was derived by Karol M., Hluchyj M., and Morgan S., 1987 [40]. Li S. Q. and Lee M. J., 1989 [41], showed that limiting throughputs of less than 58.6% occur in IQ switches given bursty arrivals with non-uniformly selected output ports. Karol M. and Hluchyj M., 1988, proposed one solution to the HOL blocking problem i.e., look-ahead queue servicing [2, 42].

2.2.2 IQ Switching

The first published description of using VOQs called "Multi-Queues" was by Tamir Y. and Frazier G., 1988 [5]. The Wave Front Arbiter (WFA), which is a sequential maximal matching algorithm, was developed by Tamir Y., and Chi H., in 1993 [6]. The Parallel Iterative Matching (PIM) concept, which is the first parallel maximal matching algorithm for IQ-VOQ switches, was developed by Anderson T., Owicki S., Saxe J., and
Thacker C., in 1993 [20]. Stiliadis D. and Varma A., 1995, discuss a variant of PIM that adds support for providing bandwidth guarantees between individual input-output pairs [43]. The 2-DRR, which is a sequential maximal matching algorithm, was developed by LaMaire R. and Serpanos D., 1994 [44]. The well-known iSLIP scheduling algorithm that improves upon PIM was developed by McKeown N., 1995. iSLIP is a maximal matching algorithm that uses round-robin counters with slip [21, 24]. McKeown N., Izzard M., Mekkittikul A. et.al, 1997, developed a VOQ crossbar-based iSLIP-scheduled packet switch that has 32-ports each operating at 10-Gbps [23].

McKeown N., Anantharam V., and Walrand J., 1996, discuss the Longest Queue First (LQF) algorithm that can achieve 100% throughput for all independent arrival processes [22]. Mekkittikul A., McKeown N., 1996, developed the Oldest Cell First (OCF) algorithm, which resolves a starvation problem caused by the LQF algorithm [38, 39]. Mekkittikul A., McKeown N., 1998, proposed the Longest Port First (LPF) algorithm and the Oldest Port First algorithm (OPF), both of which overcome the complexity problems of LQF and OCF algorithms and perform well under non-uniform traffic [45, 46].

Goudreau M., Kolliopoulos S., 2000, developed a maximal matching algorithm that uses a randomization technique to improve upon iSLIP [47]. Serpanos D. and Antoniadis P., 2000, developed a variation of iSLIP that approximates the First Come First Served (FCFS) algorithm [48]. In 2000, J. Chao proposed a scheduler that eliminates the "accept" step from the "request-grant-accept" cycle of maximal matching algorithms [49].
2.2.3 Buffered Crossbar Based Switching

The first buffered crossbar design was published by Bakka R. and Dieudonne M., 1982, [50]. Nojima S., Tsutio E., Fukuda H., and Hashimoto M., 1987, published a large multi-cabinet buffered crossbar switch implementation, which was the first published work on CICQ switching [51]. Kato Y., Shimoe T., Hajikano K., and Murakami K, 1988, gave an example of a multistage switch using (2x2) buffered crossbars [52]. Goli P. and Kumar V., 1992, published another work on a multistage crossbar switch with (2x2) buffered crossbars [53]. Another example of multistage crossbar switches with (2x2) buffered crossbars was given by Zhou B. and Atiquzzaman M., 1995 [54]. Rathgeb E., Theimer T., and Huber M., 1988, showed that a cross-point (CP) buffered switch performs better than IQ and OQ switches without speedup [55]. Gupta A., Barbosa L., and Georganas N., 1991, proposed a CICQ switch with 53-bytes CP buffers to reduce the total memory size. It was shown to achieve 87.5% throughput under uniform traffic [56].

Gupta A., Barbosa L., and Georganas N., 1992, proposed a switch with two levels of priority classes [57]. Re E. and Fantacci R., 1993, showed that CICQ switches with FIFO as well as random selection policy can approach 100% throughput [58]. Stephens D. and Zhang H., 1998, developed a buffered crossbar switch that supports Quality of Service and variable-length packets by adding schedulers at the inputs, cross points, and outputs [59]. Nabeshima M., 2000, published the first VOQ based CICQ switch [60]. Yoshigoe K. and Christensen K., 2001, proposed a Parallel-Polled Virtual Output Queued (PP-VOQ) switch with buffered crossbars that natively supports variable-length packets [11].

In 2001, Rojas-Cessa R., Oki E., Jing Z., and Chao H. J. showed that the CIXB-1 switch can achieve 100% throughput under uniform traffic [7]. Rojas-Cessa R., Oki E.,
and Chao H. J., 2001, proposed the Combined Input Cross-point Output Buffered (CIXOB-k) switch, where k is the size of the CP buffer to achieve 100% throughput under uniform and non-uniform traffic [8].

2.2.4 Parallel Packet Switches

Chiussi F., Khotimsky D., and Krishnan S., 1998 [61] is the first paper that describes the Parallel Packet Switch (PPS) concept. They describe flow level traffic distribution and aggregation. Iyer S., Awadallah A., and McKeown N., 2000 [62] is the first paper on PPS that deals with packet level traffic distribution and aggregation. The proposed PPS uses a centralized scheduler and has the same performance as that of an OQ switch. Iyer S. and McKeown N., 2001 [63] also describe a PPS architecture. The proposed PPS uses a distributed scheduler and small, high-speed buffers in the demultiplexers and multiplexers. Khotimsky D. and Krishnan S, 2001, published a paper that contains an analysis and derivation for stability criterion, speed-up, and number of planes of a PPS [64]. Mneimneh S., Sharma V., and Siu K., 2001 [65], address how to provide delay guarantees in a PPS. Aslam A. and Christensen K, 2002, provide the results of simulation study of the PPS. The proposed PPS used a distributed scheduler and virtual input queues in the multiplexer [66].

The load-balanced Birkhoff-Von Neumann switches with one-stage buffering and multi-stage buffering were introduced by Chang C. S. and Lee D.S., 2001 [67-70]. The advantage of this architecture is that it eliminates the need for a centralized scheduler. The additional issues of providing rate guarantees, maintaining packet-order, line card configuration algorithms etc., are discussed in several papers by Keslassy I., Chuang S.T., [71-73]. This architecture has been shown to be a practical way to scale Internet routers to very high capacities, and achieve throughput guarantees for all traffic patterns.
2.3 Scheduling Algorithms

2.3.1 Parallel Iterative Matching (PIM)

Parallel Iterative Matching (PIM) was developed by DEC System Research Center at the beginning of the 1990s. PIM is used in DEC's AN2, a Giga-bit switch [20]. PIM involves random access input buffers with a three phase parallel randomized scheduling algorithm, operating independently at each of the output ports. However, the use of randomizers makes PIM too slow for Tera-bit switches. The algorithm tries to find a maximum matching in several iterations. Every iteration cycle has three steps.

- **Request**: Each unmatched input sends a request to all outputs for which it has a queued cell.
- **Grant**: If an unmatched output receives any requests, it chooses one randomly to grant.
- **Accept**: If an input receives grants, it accepts one by selecting an output randomly among those outputs that granted to this input.

Tests have shown that this algorithm converges to a maximal match in \((\log N)\) iterations and has a time complexity \(O(\log N)\). Under heavy load when only a single iteration is possible this algorithm does not perform well. The throughput is limited to \((1 - 1/e)\) for large \(N\), which is approximately 63%.

2.3.2 Iterative SLIP – iSLIP

A fast and efficient algorithm, which overcomes the complexity problem of PIM and outperforms PIM in several ways, is the iSLIP algorithm. This algorithm is based on the PIM algorithm [23, 24]. The main difference is that it uses a round robin instead of a random schedule, for determining which input or output is to be matched next. This is achieved by maintaining pointers at the input and output ports. This minor variation is
shown to result in de-synchronization of the arbiters and leads to a high throughput. It is also possible to apply a prioritization scheme in order to support different classes of services.

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

- **Grant**: If an output receives many 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.

- **Accept**: If an input receives a grant, it accepts the one that appears next in a fixed round-robin schedule starting from its own output pointer. The output pointer of the round robin schedule is incremented to the next location after the accepted output.

The iSLIP algorithm has many advantages over the PIM approach. First of all, it performs better than PIM even under heavy load and is relatively easy to implement. It can achieve 100% throughput in a single iteration for uniform traffic [21, 22]. A modified form of iSLIP has been used in the Cisco 12000 router series [74]. Another benefit is that iSLIP is fairer by virtue of its round robin scheduling method and hence prevents starvation problem.

### 2.3.3 Lowest Occupancy Output First Algorithm (LOOFA)

Another algorithm called LOOFA [75] follows an iterative approach, too. It consists of the following two steps, which are performed in each cycle.

- Each unmatched input sends a request to an output with the lowest occupancy among those for which it has at least one queued cell.
• Each output, upon receiving requests from multiple inputs, selects one and sends a grant to that input.

Unlike PIM or iSLIP, this algorithm requires $O(N)$ iterations to perform correctly. The behavior under heavy load when there is only time for less number of iterations is unpredictable. This makes LOOFA unsuitable for use in high-speed implementations.

All the above are Maximum Size Matching (MSM) algorithms, which show excellent performance under uniform traffic and provide high value of instantaneous throughput. However, the performance is degraded under non-uniform traffic since they may not clear all backlogged queues. The following are the common Maximum Weight Matching algorithms (MWM), which are more complex to implement, but perform well under non-uniform traffic.

### 2.3.4 The LQF and OCF Scheduling Algorithms

If at any time slot 'n', the queue $Q_j$ is non-empty, then $L_y(n) > 0$, and there is an edge in the graph $G$, between input $i$ and output $j$. A weight $w_{ij}(n)$ is associated with each such edge. For the LQF (Longest Queue First) algorithm, $w_{ij}(n) = L_j(n)$, i.e., the queue occupancy. Queues with larger occupancies are awarded larger weights and are more likely to be served. However, LQF can lead to permanent starvation of non-empty queues, which may be of lesser lengths. The OCF (Oldest Cell First) algorithm, overcomes this problem by considering the waiting times of cells. Here, $w_{ij}(n) = W_j(n)$. Hence, cells that have been waiting for the longest time are more likely to be served [37].

### 2.3.5 The LPF and OPF Scheduling Algorithms

The Longest Port First (LPF) algorithm is a weighted algorithm giving preference to cells, based on queue occupancies. Like all maxweight algorithms, LPF determines the weights for each request and then finds a maxweight match, i.e., a match that maximizes
the total weight, \( \sum_{i,j} S_{ij}(n)w_{ij}(n) \) where, \( S_{ij}(n) \) is a service indicator- a value of one for \( S_{ij}(n) \) indicates that input ‘i’ is matched to output ‘j’. LPF gives preference to each flow, based on the degree of congestion at its input and at its output. To accomplish this, LPF uses queue occupancies to form what are called *input occupancies and output occupancies*. The input occupancy for an input port is defined as the number of cells waiting to leave the input one at a time, i.e., the sum of all queue occupancies at the input. Likewise, the output occupancy for an output port is defined as the total number of cells at all inputs competing for transmission to the output, i.e., the sum of the occupancies of all queues for the output. LPF uses input occupancies and output occupancies to determine the request weights. A request weight, \( w_{ij}(n) \), for a request from an input to output is, \( w_{ij}(n) = \begin{cases} R_i(n) + C_j(n), & L_{ij}(n) > 0 \\ 0, & otherwise \end{cases} \) where, \( R_i(n) = \sum_{j=1}^{N} L_{ij}(n) \) and \( C_j(n) = \sum_{i=1}^{N} L_{ij}(n) \) which are the row sum and column sum, of the occupancy matrix. \( L(n) \) is the occupancy matrix, each element of which is the number of cells in the VOQ \((i,j)\) at time slot ‘n’.

The *Oldest Port first* (OPF) algorithm is another maxweight algorithm. It gives preference to cells based on their waiting times instead of queue occupancies. Each request weight is a function of the waiting times of HOL cells. Like LPF, OPF does not use an individual waiting time to weight a request, but instead uses *an input waiting time* and *an output waiting time*. The waiting time of an input, \( W_{ij} \), is defined as the sum of the waiting times of all HOL cells at the input, and the waiting time of an output, \( W_{ij} \), is defined as the sum of the waiting times of all HOL cells destined for the output. In the same fashion as LPF, an input waiting time serves as a congestion indicator of cells
competing for the outgoing link of the input, and an output waiting time serves as a congestion indicator of the incoming link of the output. Accordingly, a request weight, \( w'_{i,j}(n) \), of every non-empty queue consists of two quantities as defined below.

\[
    w'_{i,j}(n) = \begin{cases} 
        R_i(n) + C_j(n), & L_y(n) > 0 \\
        0, & \text{otherwise}
    \end{cases}
\]

where, \( R_i(n) = \sum_{j=1}^{N} W'_{i,j}(n) \) and \( C_j(n) = \sum_{i=1}^{N} W'_{i,j}(n) \) are the row sum and column sum of the waiting-time matrix, \( W'_{i,j}(n) \) [38, 39].

### 2.3.6 Two Dimensional Round-Robin Scheduler - 2DRR

The basic idea behind this algorithm, indicated in Fig. 2.1, is to represent all inputs, all outputs and all requests in one matrix. This is an adjacency matrix of a bipartite graph with the input ports as one set of vertices and the output ports as the other set. The edges in this graph symbolize the requests. The algorithm starts with accepting all requests on the main diagonal and ignores all other requests, which are on the same rows and columns as the accepted requests.

This procedure continues with all other diagonals in left-to-right order until every request is accepted and a maximum matching is found. One drawback of the algorithm is that it is unfair to have a fixed left-to-right method for choosing the succeeding diagonals. A hardware implementation of the algorithm is also presented in [44].

### 2.3.7 Wave Front Arbiter (WFA)

The Wave Front Arbiter (WFA) [6] is an algorithm which is relatively simple to implement and fast enough to operate in Tera-bit switches. Similar to iSLIP, WFA also approximates a maxsize algorithm. The arbiter is made up of a two dimensional, interconnected array of simple cells.
Fig. 2.1 Two Dimensional Round-Robin Scheduler

Fig. 2.2 illustrates an example for a switch scheduled by WFA. Each cell contains a register holding a request indicator. Priority is given strictly based on cell location. Cells above have higher priority than cells below, and cells to the left have higher priority than cells to the right. Each cell has two inputs: the top input indicating that some cell above has been matched, thereby capturing the output, and the left input indicating that some cell to the left has been matched. If none of its inputs is asserted and the cell has a request, the input and the output to which the cell belonged are matched. Each cell also has two outputs: the bottom output and the right output. The bottom output is asserted when there is a matched cell above, including itself, to signal to all cells below. The right output is asserted when there is a matched cell to the left, including itself, to signal to all cells to the right.
Because of the cell dependency described above, the arbitration time is \((2N-1)T\) time units, where \(T\) is the time for each cell to assert its outputs after receiving the inputs. Implemented in 0.25 μm CMOS technology with approximately (100-200) ps propagation delay for a two-input-NAND gate, the scheduling time for a switch is approximately (12-24) ns. By wrapping around the outputs of the bottom most cells and feeding them back to the inputs of the top most cells, and similarly wrapping and feeding the rightmost cells and leftmost cells, the wrapped wave front arbiter (WWFA) [6] is able to reduce the scheduling time. The loops created by the wrapping described above are broken diagonally, allowing cell arbitration to progress in parallel like a wave front sweeping across the cell array. The broken points can be shifted every cell-time to give equal preference to all cells. However, the performance analysis shows that WWFA causes cells to experience higher mean cell delay than WFA because parallel cell arbitration sometimes makes a premature decision.

2.3.8 Neural Network Based Packet Schedulers

Hopfield neural networks have been used to find approximate maximum matching for bipartite graphs. For an N-port switch, the neural network comprises \(N^2\) neurons. The circuit is usually designed to minimize a particular energy function, which leads to the
solution of a bipartite graph-matching problem. The main problem with this method is that it requires analog circuitry. In [76], a novel combination of architecture and algorithm for a packet switch controller that incorporates an experimentally implemented optically interconnected neural network is presented.

2.3.9 Randomized Algorithms

The suite of randomized algorithms proposed in [77] is based on the principle that the state of the switch as captured by, say, its queue lengths does not change much between any two consecutive time slots. They use the matching at time $t'$ to compute the matching at time $t + 1$, eliminating the need to compute matchings from scratch, in each time slot. The algorithms are simple and performance studies have shown speed-up improvement over iSLIP and iLQF.

LAURA (*Low-complexity Algorithm Using Randomized Augmentation*) is a scheduling algorithm exploiting randomization. LAURA iteratively augments the weight of the current matching by combining its heavy edges with the heavy edges of a randomly chosen matching. It has been shown that LAURA is stable, and for small values of load, LAURA performs worse than most of the other algorithms, but this behavior is practically negligible. LAURA is a randomized algorithm, which does not necessarily find a maximal matching at lower values of load. But, for high values of load, LAURA gives very good performance.

Another randomized algorithm, SERENA (*SElf Randomized algorithm Exploiting New Arrivals*) is based on a learning scheme and the exploitation of the randomness in arrivals. For low loads, SERENA outperforms the other schemes since it is able to catch new arrivals at once and serve them. For high loads, under uniform traffic, SERENA behaves very close to LAURA.
2.3.10 The Parallel Packet Switch (PPS) Architecture

The PPS architecture has been proposed in [62-64] to emulate the behavior of an OQ switch. The architecture consists of demultiplexers, several lower speed packet switches and multiplexers, and is compared to a Clos network. It has been shown that if each of the lower speed packet switch works at an approximate rate of \(2R/k\), where \(R\) is the line rate and \(k\) is the number of center stage OQ switches, then a PPS emulates a FCFS OQ switch.

2.3.11 Deterministic Slot Allocation

The Deterministic slot allocation (DSA) algorithm is an example of how simple it is to achieve 100% throughput for uniform traffic. For an \(N \times N\) switch, DSA services each queue deterministically once every \(N\) cell-times. One possible implementation is as follows. At cell-time \(n\), input ‘i’ is matched to output \(((i+n) \mod N)\) regardless of whether the input has any cell for the output or not. For an independent identically distributed arrival process, each queue can be thought of as an M/D/1 system with a service rate of \(1/N\) of the line rate. For uniform traffic, the maximum arrival rate at any queue is always less than \(1/N\) of the line rate, and since the service rate is greater than the arrival rate, the system is considered to be stable [78].

2.4 Conclusions

The advances in optical devices and switching technology offer exciting potential for higher capacity and lower power routers, but researchers have recognized that many problems need to be addressed before these switches are implemented on a large scale. It is recognized that all packet switches must store packets, and it is not possible to store photons with current technology. Hence line card functions will continue to be electronic while optics will be used for implementing the switch fabric. Currently, researchers at
Stanford University are working on building a hybrid optical/electrical switching system. The 100 Tb/s internet router would use optical technology for a switched backplane. The router will have 625 line cards each operating at an aggregate rate of 160 Gb/s. The advances in MEMS, waveguide switches, drive electronics, direct attachment of optical modulators to silicon, optical packaging etc., suggest that it might be possible to build a high capacity low power optical switch core in the near future [16].

In this chapter, the literature on the design of scheduling algorithms for high speed switching architectures was discussed. The main focus of the study was on Input-Queued (IQ) switches, since it is the most promising architecture in terms of scalability with the speed and the number of ports. The difficulties involved in the design of high performance routers have been described. After defining the problem of scheduling as a problem of finding a match under certain constraints, the various developments in this area were introduced. The next generation of routers will still be switch based, and it is recognized that there is an overwhelming need to develop simpler and more efficient packet schedulers for high-speed switches.