ABSTRACT

Scheduling Algorithms for High Speed Switches

by
Jinhui Li

The virtual output queued (VOQ) switching architecture was adopted for high speed switch implementation owing to its scalability and high throughput. An ideal VOQ algorithm should provide Quality of Service (QoS) with low complexity. However, none of the existing algorithms can meet these requirements.

Several algorithms for VOQ switches are introduced in this dissertation in order to improve upon existing algorithms in terms of implementation or QoS features. Initially, the earliest due date first matching (EDDFM) algorithm, which is stable for both uniform and non-uniform traffic patterns, is proposed. EDDFM has lower probability of cell overdue than other existing maximum weight matching algorithms. Then, the shadow departure time algorithm (SDTA) and iterative SDTA (ISDTA) are introduced. The QoS features of SDTA and ISDTA are better than other existing algorithms with the same computational complexity. Simulations show that the performance of a VOQ switch using ISDTA with a speedup of 1.5 is similar to that of an output queued (OQ) switch in terms of cell delay and throughput. Later, the enhanced Birkhoff-von Neumann decomposition (EBVND) algorithm based on the Birkhoff-von Neumann decomposition (BVND) algorithm, which can provide rate and cell delay guarantees, is introduced. Theoretical analysis shows that the performance of EBVND is better than BVND in terms of throughput and cell delay. Finally, the maximum credit first (MCF), the Enhanced MCF (EMCF), and the iterative MCF (IMCF) algorithms are presented. These new algorithms have the similar performance as BNVD, yet are easier to implement in practice.

SCHEDULING ALGORITHMS FOR HIGH SPEED SWITCHES

by
Jinhui Li

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

May 2001

APPROVAL PAGE

SCHEDULING ALGORITHMS FOR HIGH SPEED SWITCHES

Jinhui Li

Dr. Nirwan Ansari, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Stephen Israel, Committee Member — Date
Vice President, OpenCon Systems, Inc.

Dr. Edwin Hou, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NIT

Dr. Sirin Tekinay, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering , NJIT

Dr. Symeon Papavassiliou, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering , NJIT

BIOGRAPHICAL SKETCH

Author: Jinhui Li

Degree: Doctor of Philosophy in Electrical Engineering

Date: May 2001

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ.
- Master of Science in Electrical Engineering,
  Peking University, Beijing, P. R. China, 1994.
- Bachelor of Science in Physics,
  Peking University, Beijing, P. R. China, 1991.

Major: Electrical Engineering

Presentations and Publications:

J.H. Lin, N. Ansari, and J. Li, "Nonlinear filtering by threshold decomposition," *IEEE Transactions on Image Processing,* vol. 8, no. 7, pp. 925-933, July 1999.

S. Li, J. Li, and N. Ansari, "Earliest due date first matching for input-queued cell switches," *Proc. 1999 Conference on Information Sciences and Systems,* Baltimore, MD, Mar. 1999, pp. 602-607.

J. Li and N. Ansari, "Scheduling input-queued switches by shadow departure time algorithm," *IEE Electronics Letters,* vol. 35, no. 14, pp. 1127-1128, July 1999.

J. Li and N. Ansari, "Practical Scheduling Algorithms for Input-Queued Switches," *Proc. 2000 Conference on Information Sciences and Systems,* Princeton, NJ, Mar. 2000.

J. Li and N. Ansari, "Scheduling Virtual Output Queued Switches with Low Speedup," *Proc. 2001 Conference on Information Sciences and Systems,* Baltimore, MD, Mar. 2001.

J. Li and N. Ansari, "QoS guaranteed input queued scheduling algorithms with low delay," *Proc. 2001 IEEE Workshop on High Performance Switching and Routing,* Dallas, TX, May 2001.

J. Li and N. Ansari, "Enhanced Birkhoff-von Neumann decomposition algorithm for input-queued switches," submitted to *IEE Proceedings – Communications.*

J. Li and N. Ansari, "The Iterative Shadow Departure Time Algorithm," submitted to *Journal of Computer Networks.*

To my beloved family

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Professor Nirwan Ansari, who not only served as my research supervisor, contributing valuable resources, insight, and intuition, but also constantly gave me support, encouragement, and guidance in various ways to make this dissertation possible. Special thanks are given to Dr. Stephen Israel, Dr. Edwin Hou, Dr. Sirin Tekinay, and Dr. Symeon Papavassiliou for serving on my dissertation committee.

I wish to express thanks for Shizhao Li's help and advice. I am grateful to Mark Simkins, Ambalavanar Arulambalam, Jianguo Chen, Pingping Zong, and Kun Wang for the helpful discussions and suggestions. I would like to thank Lucent Technologies, the New Jersey Commission on Science and Technology, and the New Jersey Commission on Higher Education for supporting of my research. I would like to give thanks to Ronald S. Kane, Clarisa González-Lenahan, Jeffrey Grundy, and Vicky McPhillip for their assistance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

LIST OF ACRONYMS

2DRR:	Two-Dimensional Round-Robin
BVND:	Birkhoff-von Neumann Decomposition algorithm
CCF:	Critical Cells First algorithm
CIOQ:	Combined Input Output Queueing
CLR:	Cell Loss Ratio
CTD:	Cell Transfer Delay
EBVND:	Enhanced BVND algorithm
EDDFM:	Earliest Due Date First Matching algorithm
EMCF:	Enhanced MCF algorithm
EPIM:	Enhanced PIM algorithm
FIFO:	First-In First-Out
FIRM:	FCFS in Round Robin Matching algorithm
GSA:	Gale-Shapley Algorithm
GS-LQF:	Gale-Shapley LQF algorithm
GS-OCF:	Gale-Shapley OCF algorithm
HOL:	Head-of-line
HTA:	Home Territory Algorithm
IQ:	Input Queueing
IRRM-MC:	Iterative Round Robin with Multiple Classes
ISDTA:	Iterative SDTA
iLPF:	Iterative LPF algorithm
i-LQF:	Iterative LQF algorithm
IMCF:	Iterative MCF algorithm
i-OCF:	Iterative OCF algorithm
iSLIP:	iSLIP algorithm
LQF:	Longest Queue First algorithm
LNQF:	Longest Normalized Queue First algorithm
LOOFA:	Lowest Occupancy Output First Algorithm
LOOTFA:	Lowest Output Occupancy and Timestamp First Algorithm
LPF:	Longest Port First algorithm
LRU:	Least-Recent Used algorithm
MCF:	Maximum Credit First algorithm
MMDP:	Markov Modulated Deterministic Process
MSM:	Maximum Size Match
MUCF:	Most Urgent Cell First algorithm
MUCS:	Matrix Unit Cell Scheduler
MWM:	Maximum Weight Match
OQ:	Output Queueing
PGPS:	Packetized Generalized Processor Sharing

PIFO:	Push-In First-Out
PIM:	Parallel Iterative Matching algorithm
RRM:	Round Robin Matching algorithm
QoS:	Quality of Service
RPA:	Reservation with Preemption and Acknowledgement
SDT:	Shadow Departure Time
SDTA:	Shadow Departure Time Algorithm
SIMP:	Successive Incremental Matching over Multiple Ports
SPIM:	Simplified PIM algorithm
SSF:	Store-Sort-and-Forward algorithm
VOQ:	Virtual Output Queueing
WFA:	Wave Front Arbitration
WFQ:	Weighted Fair Queueing
WF^2Q:	Worst-case Fair Weighted Fair Queueing
WFBVND:	Wave Front BVND algorithm
WFBVND-$logN$:	WFBVND with $logN$ iterations
WPIM:	Weighted PIM algorithm
WRFA:	Weighted Rate Filling algorithm
WWFA:	Wrapped Wave Front Arbitration

CHAPTER 1

INTRODUCTION

1.1 Input Queueing Versus Output Queueing

There are two basic types of switching architectures: output queued (OQ) switching architecture and input queued (IQ) switching architecture. When a packet reaches an OQ switch (see Fig. 1.1) it will be queued in its output queue immediately. The packet will be staying in the output queue until it is transmitted from the switch, and thus 100% throughput can be achieved. OQ switches can provide quality of service (QoS) guarantees by using the scheduling mechanisms [69] such as weighted fair queueing (WFQ) [15] (or packetized generalized processor sharing (PGPS) [55][56]) and worst-case fair weighted fair queueing (WF^2Q) [3]. However, one problem of the OQ switch is that the fabric of an $N \times N$ OQ switch must run N times as fast as its line rate, i.e., the speedup of an OQ switch is N. The bandwidth of the memory of such a switch should be $N + 1$ times of the line rate because the memory should be written and read during the same timeslot.

When a packet reaches an IQ switch (see Fig. 1.2), it is placed in its input queue until it can be scheduled across the fabric. In the IQ switch, the packet can be transmitted out of the switch immediately when it comes to the output port. The fabric of an IQ switch is only required to run as fast as the line rate and the memory needs to run twice as fast, i.e., the speedup of an IQ switch is 1.

In the high speed networks, the memory and fabric with a bandwidth that are larger or at least equal to N times that of line rate could be unavailable. Therefore, the IQ switching architecture was adopted for high speed switch implementation owing to its scalability. One of the major problems with the IQ switching architecture is the head-of-line (HOL) blocking: the HOL cell, which cannot be forwarded because of output contention, can block the output-contention-free cells in the same queue when FIFO is used. HOL blocking limits the throughput of the IQ switch using a single FIFO queue in each input to approximately $2 - \sqrt{2} \approx 58.6\%$ under *i.i.d.* Bernoulli traffic when N is large [29]. The situation is even worse under bursty traffic: the maximum throughput of such a switch decreases monotonically with the burstiness of traffic and reaches to 50% when burstiness is large [36]. Stationary blocking is another problem of FIFO IQ switches, where the total throughput of the switch can be as little as the throughput of a single link under certain periodic traffic even when N is very large [37].

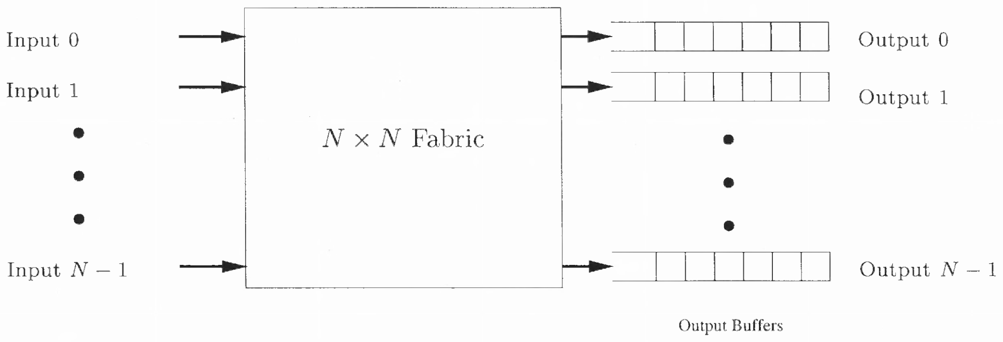


Figure 1.1 Output queued switching architecture

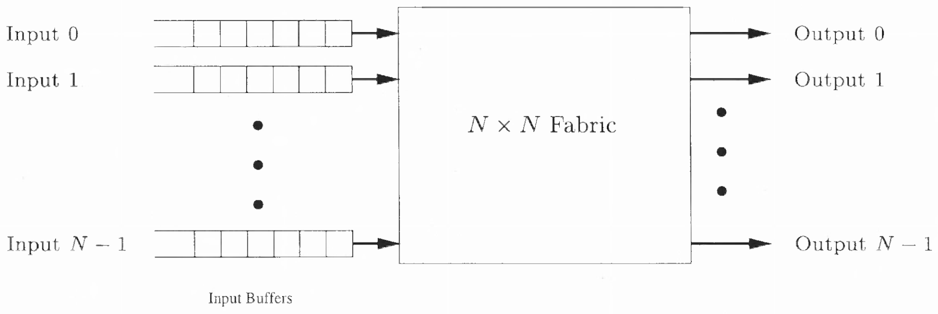


Figure 1.2 Input queued switching architecture

HOL blocking can be partly reduced by increasing the speed of the fabric. The ratio of bandwidth between fabric and input link is defined as speedup. When the speedup is larger than 1 and smaller than N, buffers are required at the outputs as well as inputs. This switching architecture with both input and output buffering is called combined input output queued (CIOQ) switch (see Fig. 1.3). Chuang *et al.*

[12] proved that a CIOQ switch with a speedup of two can exactly mimic an OQ switch.

Previous research [1][43][48][49] shows that HOL blocking can also be completely eliminated in IQ switches by adopting virtual output queueing (VOQ) [64][1] without a speedup, in which multiple VOQs directed to different outputs are maintained at each input as shown in Fig. 1.4. The throughput of an VOQ switch can be increased to 100% under all kinds of admissible independent traffic by using well-designed scheduling algorithms. VOQ architecture can also be used in CIOQ switch which has a speedup as shown in Fig. 1.5.

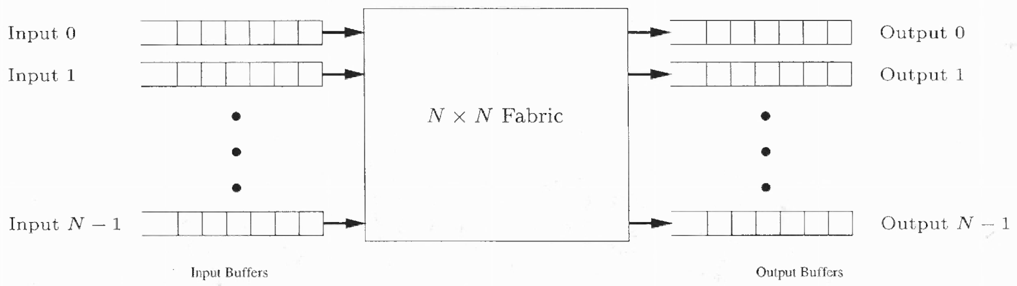


Figure 1.3 Combined input output queued switching architecture

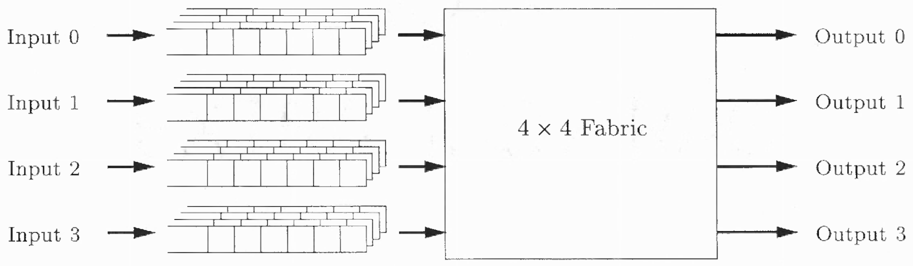


Figure 1.4 Virtual output queued switching architecture without speedup

1.2 Our Switch Model and Traffic Model

Consider an $N \times N$ input-queued switch consisting of N inputs, N outputs, and a non-blocking switch fabric such as crossbar. The packets, which may have variable lengths, are broken into fixed length cells when they arrive in the inputs. After the

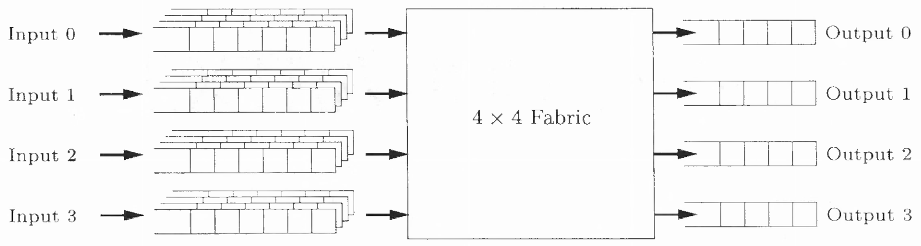


Figure 1.5 Virtual output queued switching architecture with speedup

cells cross the fabric, they are reassembled to the original variable length packets. In order to eliminate the HOL blocking, VOQ is adopted in this architecture. The VOQ directed to output j at input i is denoted by $Q_{i,j}$. If $Q_{i,j}$ is not empty, there will be a request from input i to output j. The fundamental objective of scheduling an VOQ switch is to find a contention free match based on the connection requests, i.e., at most one input can be matched to each individual output, and vice versa. Let $\mathbf{S} = (S_{i,j})$ be the matching matrix, which indicates the match between inputs and outputs. If input i and output j are matched, then $S_{i,j} = 1$; otherwise, $S_{i,j} = 0$. At the end of the timeslot, a cell is transmitted from input i to output j if $S_{i,j} = 1$ and $Q_{i,j}$ is not empty.

Assume every input and output of an $N \times N$ CIOQ switch have the same line rate. Timeslot is defined as the time required to transmit a cell with the line rate. The time axis is divided into frames [20], where each frame contains exactly F timeslots. In every frame, it is assumed that the fabric can make X transmissions. Fabric slot is defined as the time required to make one transmission by the fabric. In each fabric slot, at most one cell can be removed from each input and at most one cell can be sent to each output. Thus, speedup, ω is equal to X/F. An example of $F = 3$, $X = 4$, and $\omega = 4/3$ is shown in Fig. 1.6. Logically, one frame can be divided into $2F + X$ phases: F input phases, X transmission phases, and F output phases, as shown in Fig. 1.7. In each input phase, at most one cell can arrive in

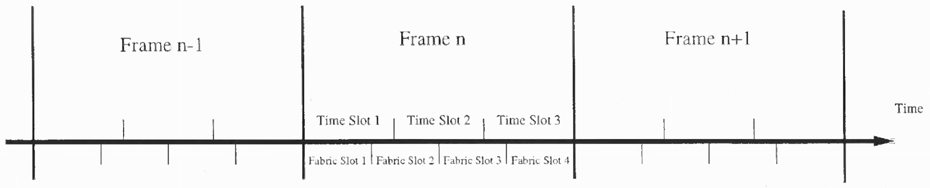


Figure 1.6 Illustration of frames with $F = 3$, $X = 4$, and speedup $\omega = F/X = 4/3$.

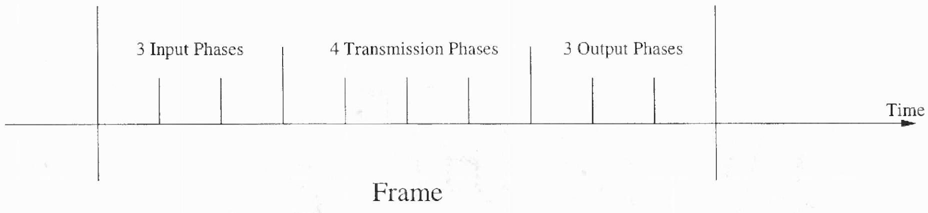
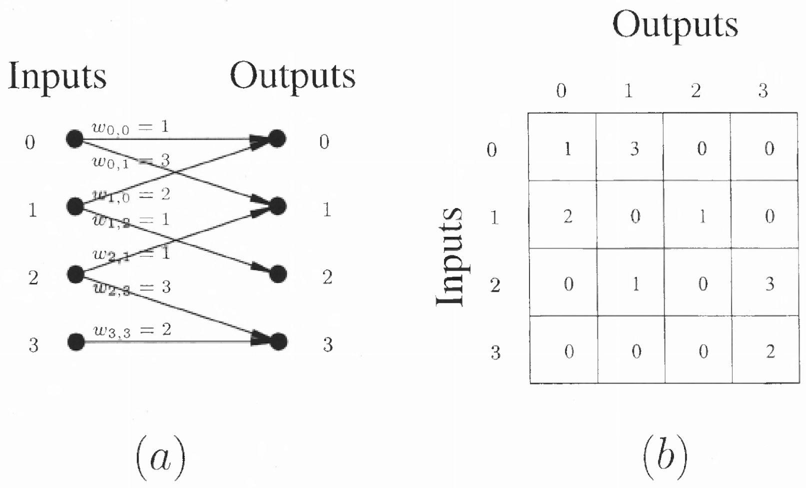


Figure 1.7 Logical phases in a frame with $F = 3$ and $X = 4$.

each input. Likewise, in each output phase at most one cell can depart from each output. Cells in inputs are transmitted to outputs during transmission phases. In each transmission phase, at most one cell can be removed from each input and at most one cell can be transmitted to each output.

Finding a contention-free match between inputs and outputs is equivalent to solving a bipartite graph matching [1][66] problem as shown in Fig. 1.8(a). Each vertex on the left side of Fig. 1.8(a) represents an input and that on the right side represents an output. An edge connects input vertex i and output vertex j if $Q_{i,j}$ is not empty. Each VOQ $Q_{i,j}$ can be associated with a weight $w_{i,j}$, which is defined differently by different algorithms. For example, $w_{i,j}$ can be set to $L_{i,j}$ or $\varpi_{i,j}$, where $\mathrm{L}_{i,j}$ is the length of $Q_{i,j}$ and $\varpi_{i,j}$ is the waiting time of the HOL cell of $Q_{i,j}$. This problem can also be expressed in matrix form by showing the $N \times N$ VOQs as in Fig. 1.8(b). Every block in Fig. 1.8(b) represents a VOQ and the number inside the square box is the weight of the VOQ.



	0	1	2	3
0	1	3	0	0
1	2	0	1	0
2	0	1	0	3
3	0	0	0	2

Figure 1.8 A bipartite graph matching example: (a) the request graph, and (b) the VOQs.

The major task of the VOQ scheduling algorithms is to find out the solution of the bipartite graph matching problem. Note that the solution may not be unique. A maximum size match (MSM) maximizes the total number of unique pairings. The best known algorithm to compute the MSM has the complexity of $O(N^{2.5})$ [25]. The MSM of Fig. 1.8 is depicted in Fig. 1.9(a), where the matched VOQs are shown in thick border boxes. A maximum weight match (MWM) has the maximum aggregate weight, i.e., $\mathbf{S} = \arg\max_{\mathbf{S}}[\sum_{i,j} S_{i,j} w_{i,j}]$. The complexity to compute the MWM is $O(N^3)$ [66]. Figure 1.9(b) is a MWM solution of Fig. 1.8. The MSM is a special case of the MWM with the weights of the non-empty VOQs set to 1 and those of empty VOQs set to 0, respectively. Figure 1.9(c) shows a maximal match which means that no pair can be trivially added without alternating the current connections. A MSM or MWM is always a maximal match, but the reverse may not be true. Different from the above matches, the stable marriage match [19] seeks to match N inputs with N outputs so that there is no pair consisting of an input and an output which prefer each other to the "partners" with which they are currently matched. If input i prefers output j with a degree of $w_{i,j}$, and output j also prefers input i with the same degree of $w_{i,j}$, then, Fig. 1.9(d) is the stable marriage matching solution of Fig. 1.8. In Fig. 1.9(d), the input-output pair $(3, 2)$, which has no cell to send, will be ignored when the cells are forwarded. Stable marriage matching problem can be solved by Gale-Shapley Algorithm (GSA) in $O(N^2)$ running time [19].

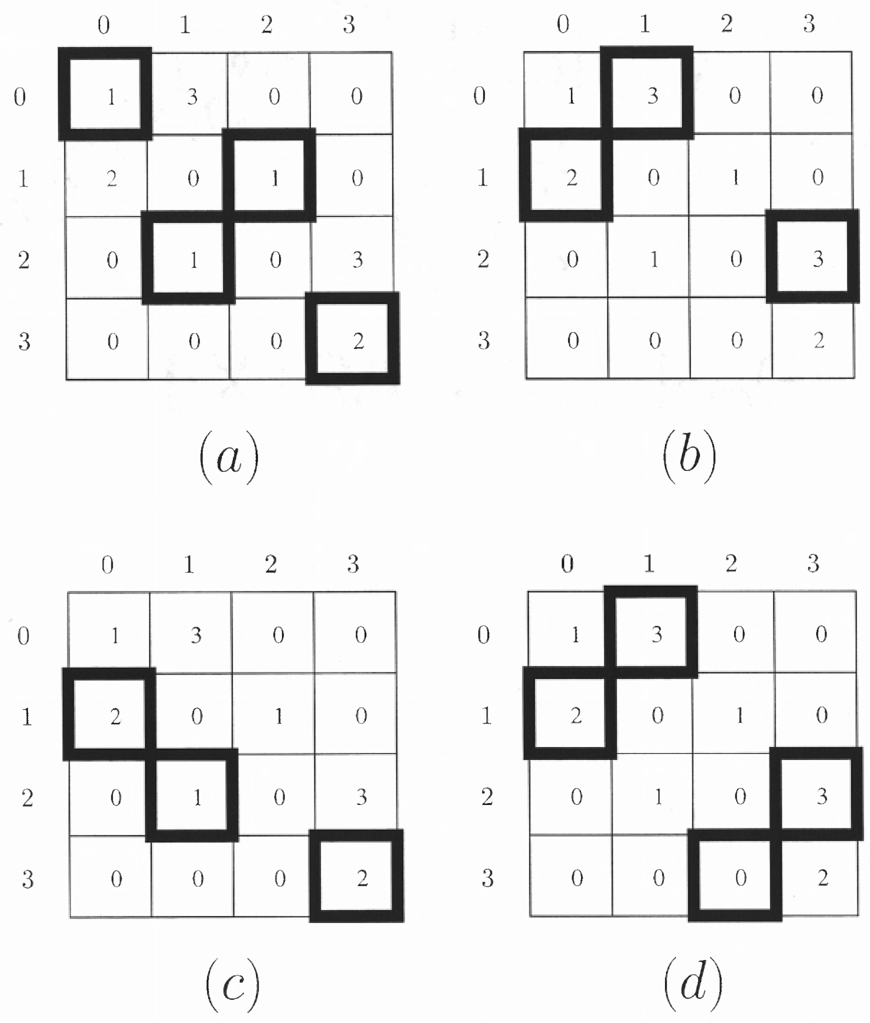


Figure 1.9 Solutions of the bipartite graph matching problem: (a) a maximum size match, (b) a maximum weight match, (c) a maximal match, and (d) a stable marriage match.

The traffic in a real network is highly correlated from cell to cell while cells tend to arrive at the switch in "bursts". One way of modeling a bursty source is using an $on-off$ model in the discrete-time domain. This model is equivalent to a two-state Markov modulated deterministic process (MMDP)[4]. These two states, OFF state and ON state, are shown in the Fig. 1.10. In the OFF state, the source does not send any cells. In the ON state, the source sends data cells at the peak cell rate (r_p). The source can shift from one state to another as shown in Fig. 1.10. In the discrete-time domain, state changes may occur only at the end of a time-slot. At each timeslot, the source in the OFF state changes to the ON state with a probability α. Similarly, the source in the ON state changes to the OFF state with a probability β. Note that there is no correlation between the two probabilities. The probabilities of the source being in the OFF state and ON state are given by $\Psi_{off} = \frac{\beta}{(\alpha+\beta)}$ and $\Psi_{on} = \frac{\alpha}{(\alpha+\beta)}$, respectively. The bursty source is characterized by the peak cell rate (r_p), the average cell rate (r), and the average number of cells per burst (B). The burstiness of the traffic is defined as the ratio of the peak cell rate and average cell rate. Given these parameters, the state transition probabilities can be computed as $\alpha = \frac{r}{B(r_p - r)}$ and $\beta = \frac{1}{B}$.

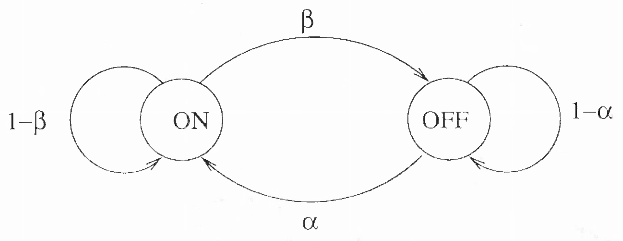


Figure 1.10 Simple $on - off$ traffic model

CHAPTER 2

EXISTING SCHEDULING ALGORITHMS

2.1 Existing Input Queued Algorithms

It has been shown by simulations [42] that when arrivals are uniform and independent, MSM can provide 100% throughput. However, when arrivals are non-uniform, the performance of this algorithm is not good. If the traffic is admissible, it can lead to unfairness and instability [43]. If the traffic is inadmissible, it can lead to starvation under certain conditions [1]. Owing to the limitations of MSM, scheduling algorithms adopt MWM, stable match, maximal match, and other solutions of the bipartite graph matching problem.

2.1.1 Parallel Iterative Matching (PIM) Algorithm and Its Variations

Early IQ algorithms such as parallel iterative matching (PIM) [1], two-dimensional round-robin (2DRR) [33], and wave front arbitration (WFA) [65] tried to find the maximal matching solution of the bipartite problem because it has a lower computational complexity than the MWM and MSM. PIM is a randomized parallel algorithm developed by Digital's Systems Research Center for the AN2 switch, which is a 16×16 crossbar switch that has the line rate of $1Gbps$ [1]. This switch was commercialized as the Gigaswitch/ATM.

PIM was designed to find a maximal bipartite match by iterations. It iterates the following three steps:

1. *Request:* Each unmatched input i sends a request to each output j if $Q_{i,j}$ is not empty.

2. *Grant:* If an unmatched output j receives any requests, it chooses one randomly to grant.

3. *Accept:* If an input i receives any grants, it chooses one randomly to accept.

Repeating the above iteration, PIM will find a maximal match eventually. However, in the worst case, it will take N rounds to converge to the maximal solution. It can be proved that on average the algorithm can match at least $\frac{3}{4}$ of the remaining possible pairs and it can find a maximal match in $O(log_2N)$ iterations on average [1]. Simulations show, using PIM, maximal matching can be reached in 4 iterations for 16×16 switch over 99% of the time [1] and asymptotic 100% throughput can be achieved when sufficient number of iterations are used [42]. Recently, reference [53] gives a closed-form solution of the maximum throughput of switches using PIM algorithm under *i.i.d.* Bernoulli traffic when the traffic load is symmetric. The analytical result shows when N is 8 or 16, high throughput ($> 90\%$) can be achieved in 3 iterations [53]. Recently, Dai and Prabhakar [14] proved that with a speedup of two the maximal algorithms such as PIM can achieve 100% throughput under arbitrary arrival patterns. Since no central arbitrator is used in PIM, it is easier to be implemented in high speed.

One of the problems of PIM is that it is unable to allocate bandwidth flexibly among the connections. Furthermore, PIM is unfair: it cannot allocate the bandwidth fairly among the competing connections [1]. Since contention exists at both inputs and outputs, PIM will give lower bandwidth to connections that have more contending connections. Statistical matching [1], a generalization of PIM, was introduced to overcome the unfairness of PIM. An iteration of Statistical Matching is composed of two steps: grant and accept, in which grant is initiated by outputs. By using Statistical Matching, bandwidth can be allocated according to demands. Yet, it limits the maximum throughput to $(1 - 1/e) \times (1 + 1/e^2) \approx 72\%$ [1].

Another problem of PIM is its complexity. Since, randomness is used in PIM, it is difficult and costly to implement when the line rate is high [41]. Round-robin matching (RRM) algorithm [41][42][46], a simplified version of PIM, is introduced to solve the problem of complexity by using round-robin arbitration. RRM consists of the following three steps:

1. *Request:* Each unmatched input i sends a request to each output j if $Q_{i,j}$ is not empty.

2. *Grant:* If an unmatched output j receives any requests, it chooses the input with the highest priority from all the inputs that has a request. In every output j, the priority of input i equals to $(i-g_j) \bmod N$, where g_j is the highest priority pointer, and $i, j = 1, 2, ..., N$. Then pointer g_j increases (modulo N) by one.

3. *Accept:* If an input i receives any grants, it chooses the output with the highest priority. In every input i, the priority of output j equals to $(j - a_i) \bmod N$, where a_i is the highest priority pointer. Then, pointer a_i increases (modulo N) by one.

The problem of RRM is that it does not perform well in terms of throughput, which suffers from blocking. Just like PIM with a single iteration, the maximum throughput of RRM is limited to $(\frac{N-1}{N})^N$ under *i.i.d.* Bernoulli traffic. When N is large, the maximum throughput tends to $1 - 1/e \approx 63\%$ [46].

With the objective of improving the performance of RRM, iSLIP [41][42][46] algorithm was introduced. iSLIP has the same steps as RRM except for the Grant step:

2 *Grant:* If an unmatched output j receives any requests, it chooses the input with the highest priority from all the inputs that has a request. In every output j, the priority of input i equals to $(i-g_j) \bmod N$, where g_j is the highest priority pointer, and $i, j = 1, 2, ..., N$. Then, pointer g_j increases (modulo N) by one if and only if this grant is accepted.

This small change makes iSLIP performs much better than RRM. Simulations [42][46] show that iSLIP can achieve asymptotic 100% throughput in just one iteration.

Using iSLIP, a non-empty VOQ can always be served in less than N^2 timeslots [46]. Therefore, no connection will be starved by this algorithm. When multiple iterations are used, similar to PIM, iSLIP needs N iterations to converge to the maximal match in the worst case, but only need $log_2 N$ iterations on average [46].

iSLIP has several variations. Prioritized iSLIP [46] maintains a separate FIFO for every priority level p from every input i to every output j. If the total number of priority levels is P, then Prioritized iSLIP will maintain $P \times N \times N$ separate FIFOs. Using this algorithm, the traffic with higher priority will get better service. Also, threshold iSLIP and weighted iSLIP are introduced in [46]

The least-recently used (LRU) [42][45] algorithm is another variation of iSLIP. This algorithm gives the highest priority to the least recently used port and the lowest priority to the most recently used port. Simulations [45] show that both LRU and iSLIP can reduce burstiness of traffic at the output of switch. Reference [45] also suggests that the performance of LRU is not better than iSLIP, though its complexity is higher. The iterative round robin with multiple classes (IRRM-MC) [51][52] algorithm is a prioritized version of iSLIP. Separate queues allocated for different priority classes, IRRM-MC runs different iteration for different class beginning from the highest priority. By discriminating priority classes, IRRM-MC provides better QoS for the flows with higher priority.

When the simplified PIM (SPIM) [50][51] is used, an input only sends at most one request to the output in each iteration. Thus, the accept step can be erased since one input can get at most one grant. In the third step, the output notifies all the inputs whether the output is matched or not. Then, in the next iteration, the unmatched inputs will only send request to the unmatched output. These modification makes SPIM easier to implement than PIM, yet the throughput performance of SPIM is almost the same as PIM [50].

The weighted PIM (WPIM) [61] is introduced aiming to provide bandwidth guarantee in IQ switch. WPIM divides the time axis into frames, in which each frame consists of F timeslots, where F is an integer. WPIM seeks to guarantee that on average at least $c_{i,j}$, the credit of VOQ $Q_{i,j}$, cells can be transmitted from $Q_{i,j}$ in each frame. It iterates the following four steps:

1. *Request:* Each unmatched input i sends a request to each output j if $Q_{i,j}$ is not empty.

2. *Mask:* Every VOQ $Q_{i,j}$ has a mask bit $m_{i,j}$. $m_{i,j}$ is set to 1, if the total number of cells in $Q_{i,j}$ that have transmitted is larger or equal to its credit in current frame. Otherwise, $m_{i,j}$ is set to 0. The masked VOQs are ignored by the outputs.

3. *Grant:* If an unmatched output j receives any unmasked requests, it chooses one randomly to grant.

4. *Accept:* If an input i receives a grant, it chooses one randomly to accept. The matched input-output pairs can be removed from subsequent iterations.

Compared to Statistical Matching, WPIM is more flexible in bandwidth allocation, and can provide higher bandwidth utilization [61].

The enhanced PIM (EPIM) [38] has also Request, Grant and Accept steps in each iteration. EPIM schedules future timeslot as well as current timeslot by enabling an input to accept multiple grants belonging to different timeslots. By introducing more complexity than PIM, EPIM can achieve maximal match in less iterations than PIM. For example, the delay performance of an 16×16 switch using EPIM with 2 iterations is shown by simulations to be as good as PIM with 4 iterations [38].

FCFS in round robin matching (FIRM) [60] algorithm improved iSLIP by modifying the Grant step:

2 *Grant:* If an unmatched output j receives any requests, it chooses the input with the highest priority from all the inputs that has a request. In every output j, the priority of input i equals to $(i - g_j) \bmod N$, where g_j is the highest priority pointer and $i, j = 1, 2, ..., N$. If this grant is accepted, g_j increases (modulo N) by one; otherwise, g_j is set to the granted input.

Serpanos and Antoniadis [60] proved that, in the worst case, a request will wait for $N^2 + (N-1)^2$ timeslots to be served using iSLIP. On the other hand, a request can be served in N^2 timeslots using FIRM. They demonstrated by simulations [60] that the average cell delay of FIRM would be improved about 50% comparing to iSLIP when the traffic load is above 95%.

2.1.2 Maximum Weight Matching Scheduling Algorithms and Their Variations

A resource allocation model and scheduling method were introduced in [67] to achieve maximum throughput. It proves that the system is stable under the policy that finds the optimal solution, and the problem of scheduling IQ switches is a special case of it. Later, this result was independently discovered by McKeown *et al* in [43], and the algorithm is referred to the longest queue first (LQF) [43][47]. LQF sets $w_{i,j}$, the weight of VOQ $Q_{i,j}$, to be its queue length $L_{i,j}$, and finds the MWM. They [43] proved, under any admissible and independent arrival processes, LQF is stable, i.e., $E[L_{i,j}] < \infty, \forall i, j$, which implies that LQF can achieve 100% throughput. Later, Dai and Prabhakar [14] lifted the *i.i.d.* assumption of the traffic in [67] and [43]. Using fluid model techniques, they proved that LQF can achieve 100% throughput under arbitrarily distributed traffic patterns if only the input traffic is admissible and obeys the strong law of large numbers [14].

The iterative LQF (i-LQF) [42], a variation of LQF, was introduced as a simplification. Using iterative algorithm instead of MWM, i-LQF has lower computational complexity. If sufficient iterations can be completed, i-LQF can reach the maximal match. GS-LQF [42] is the approximation of LQF. It uses GSA to find the stable match. Simulations [42] show that the performance of GS-LQF is identical to i-LQF. The longest normalized queue first (LNQF) [34] is another variation of LQF. It sets $w_{i,j}$ to be the normalized queue length, which is the total queue length of the VOQ divided by its rate. LNQF has better performance than LQF in terms of fairness and burstiness reduction.

The oldest cell first (OCF) [42][47] sets $w_{i,j}$ to $\varpi_{i,j}$, the waiting time of the HOL cell in $Q_{i,j}$, and finds the MWM. OCF is stable and starvation-free under all independent and admissible traffic [42]. Also, OCF has an iterative version — i-OCF [42], which uses an iterative procedure to find the maximal match. GS-OCF, the stable marriage matching approximation of OCF, is also introduced in [42]. Simulations suggest that the performance of GS-OCF and i-OCF are identical.

The longest port first (LPF) [49] is proposed to overcome the complexity of LQF. LPF sets $w_{i,j}$ to be a function of the length of $Q_{i,j}$.

$$w_{i,j} = \begin{cases} \sum_{j'} L_{i,j'} + \sum_{i'} L_{i',j}, & L_{i,j} > 0 \\ 0, & \text{else} \end{cases} . \tag{2.1}$$

Using a modified Edmonds-Karp maximum size matching algorithm [13][66], LPF finds a match which is both MSM and MWM with the complexity of $O(N^{2.5})$ [49]. LPF can achieve 100% throughput under both uniform and non-uniform traffic. The iterative LPF (iLPF) [49] is the iterative version of LPF.

The weighted arbitration [59] algorithm was introduced to support priorities in weighted matching by giving the cells of higher priority level a larger weight. Also, the successive incremental matching over multiple ports (SIMP) algorithm is introduced in [59] which is an approximation of MWM. Note that all the algorithms that use MWM can also use SIMP which has the complexity of $O(N^2)$.

The shakeup technique [22] is a randomized approach that can be used in conjunction with a number of existing weighted and unweighted heuristics to substantially improve solution quality. Unweighted shakeup randomly selects an unmatched

Table 2.1 MWM scheduling algorithms and their variations

Algorithm	Weight	Match	Complexity
LQF	$L_{i,j}$	MWM	$O(N^3)$
GS-LQF	$L_{i,j}$	Stable	$O(N^2)$
i-LQF	$L_{i,j}$	Maximal	$O(N^2)$
LNQF	$L_{i,j}/r_{i,j}$	MWM	$O(N^3)$
OCF	$\varpi_{i,j}$	MWM	$O(N^3)$
GS-OCF	$\varpi_{i,j}$	Stable	$O(N^2)$
i-OCF	$\varpi_{i,j}$	Maximal	$O(N^2)$
LPF	$\begin{cases} \sum_{j'} L_{i,j'} + \sum_{i'} L_{i',j}, & L_{i,j} > 0 \\ 0, & \text{else} \end{cases}$	MSM	$O(N^{2.5})$

port from an initial match, then it matches this port even it will break an existing pair. The return of the unweighted shakeup is not less than the initial match. On the other hand, weighted shakeup favors the ports with heavy load. Combining the existing heuristics and shakeup will let to better stability and cell delay.

2.1.3 Algorithms with Delay Bound

Cell delay guarantee is important for real-time applications. Unfortunately, none of the IQ algorithms mentioned above can provide this guarantee. Therefore, some algorithms were proposed for this purpose.

The Slepian-Duguid [1] algorithm divides the time axis into frames such as WPIM, where each frame consists of F timeslots, in which F is a fixed number. CBR traffic allocates the bandwidth, and then a fixed schedule can be computed. Slepian-Duguid theorem [26] suggests that a schedule can always be found if the total number of cells to any output is not larger than F. The slots that are not used by CBR traffic can be filled by the best-effort traffic. Using Slepian-Duguid algorithm cell delay is bounded by $2F$ at every switch if the switches are synchronized as in the telephone network. Reference [1] shows that when the switches are not synchronized the delay bound is about four or five frâmes for local area networks.

The store-sort-and-forward (SSF) [35] algorithm is similar to the Slepian-Duguid algorithm: cells arriving during a frame are first held in the input buffers and are then sorted-and-transmitted within the next frame. The difference is that SSF needs to compute the schedule in every frame. When the traffic conforms to the (r, T) model [20], SSF can guarantee delay bound and 100% throughput. Both of the Slepian-Duguid and SSF algorithms have the problem of the rate granularity limitation: smaller F leads to lower cell delay but coarser granularity of bandwidth allocation. The traffic constraint of SSF is not as tight as the Slepian-Guguid algorithm, but its computational complexity is higher.

Chang *et al.* proposed [6][7] the Birkhoff-von Neumann decomposition (BVND) algorithm which is based on a decomposition result by Birkhoff and von Neumann for a doubly stochastic matrix. This algorithm can provide 100% throughput for all non-uniform traffic. Furthermore, if the traffic is (σ, r)-upper constrained, cell delay can be deterministically guaranteed using this algorithm.

A class of algorithms with $O(N^2)$ complexity is proposed by Kam and Siu by setting the weight of $Q_{i,j}$ to be the functions of its queue length, the largest waiting time of the cells, and the outstanding credit, and then finding the stable marriage match [28]. They proved that some of these algorithms can support up to 50% bandwidth reservation with constant delay bounds. As their proof is based upon Lyapunov technique which gives very loose bounds, they showed by simulations that the bandwidth reservation can be up to 90%.

2.1.4 Other Input Queued Algorithms

Wave front arbitration (WFA), wrapped wave front arbitration (WWFA), and several other algorithms are proposed in [65]. WFA and WWFA can achieve high throughput and can be readily realized in hardware. In every timeslot, these algorithms scan the VOQs and find the maximal match by $2N-1$ and N steps, respectively. In the mth step, WFA checks the VOQs that satisfy $i+j=m$, where $0 \leq m < 2N$. If a VOQ is not empty and its input and output are both available, then this VOQ is added to the match. WWFA checks the wrapped diagonal $(i+j) modN = m$, where $0 \leq m < N$, so that it can reach the maximal match in just N steps.

Two-dimensional round-robin (2DRR) [33] algorithm is another algorithm that uses the round-robin approach to achieve the maximal match. Self-firing algorithm introduced in [32] is a variation of 2DRR. In [63], the concept of tracking policies for fluid policies is extended to the IQ switches. It shows that the tracking policy always exists for the switch size of 2×2. Heuristic tracking policy is provided for general case of $N \times N$ switches in [63]. Reservation with preemption and acknowledgment (RPA) [39][40] is a quasi-optimal algorithm with the complexity of $O(N^2)$ based on reservation rounds and an acknowledgment round. RPA can also deal with multiple traffic classes [40].

Matrix unit cell scheduler (MUCS) [17][18] demonstrates good performance under different traffic scenarios. It is easy to be implemented in hardware because of its low interconnect and transistor count. MUSC sets the heaviest weight to the VOQ that has the least contention in the same row and the same column, and then find the stable marriage match. Simulations show that MUCS can achieve nearly 100% throughput with simple hardware implementation.

Reference [5] researched the benefit of an IQ switch to use the future information. They stated that a 2×2 IQ switch can be equivalent to a 2×2 OQ switch when future information is available. However, the equivalence is not held for larger switches. It is NP-hard to use future information optimally.

2.2 Existing Combined Input Output Queued Algorithms

Another approach to achieve QoS guarantees is to increase the speed of the fabric. The speedup of the switch, ω, is defined as the ratio of the bandwidth between fabric and input link. When ω is larger than 1 and smaller than N, buffers are required at the outputs as well as inputs. Early effort on CIOQ switch can be found in [27][24][68][54][10][8]. [68] introduced knockout switch instead of speedup, in which they employed a parallelism factor K. A knockout switch with a parallelism factor of K is different from a switch with a speedup of K. In the knockout switch, the K packets to an output during the same timeslot cannot come from the same input. [54] and [10] show that with $K = 4$, throughput can achieve more than 99.5% under the *i.i.d.* traffic even when N is very large.

Guerin *et al.* [23][16] suggested that a CIOQ switch with a speedup less than two is sufficient to achieve 100% throughput. They also demonstrated by simulations that a speedup of two is sufficient to achieve nearly the same delay performance as that of an OQ switch even for bursty traffic. In [44], Mckeown *et al.* proved that employing the home territory algorithm (HTA) and VOQ, a CIOQ switch with a speedup greater than $N/2$ is always work-conserving. Later, Prabhakar and McKeown showed [57] that using most urgent cell first (MUCF) algorithm, a CIOQ switch of any size with VOQs and speedup of 4 can perform identically to an FIFO-OQ switch under arbitrary input traffic patterns. Then, Krishna *et al.* [30] proposed a simpler algorithm, which can emulate FIFO-OQ switch with a speedup of 3. Charny *et al.* [9] showed that with any maximal matching algorithm, the CIOQ switch with the speedup of four is sufficient to ensure 100% asymptotic throughput. They also showed that delay guarantees can be achieved with a relatively simple algorithm and speedup independent of the switch size. Instead of trying to emulate OQ switch, Krishna *et al.* [31] showed that using lowest occupancy output first algorithm (LOOFA), a CIOQ crossbar switch with a speedup of two is work-conserving, thus providing the same throughput performance as an OQ switch. Then Rodeheffer and Saxe [58] showed that using the lowest output occupancy and timestamp first algorithm (LOOTFA), a refinement of LOOFA, with a speedup of at least 3, the LOOTFA crossbar switch is both work-conserving and order-conserving. Reference [62] tries to prove that a CIOQ with a speedup of two can exactly emulate an OQ switch, but later it was pointed out that the algorithm and proofs are not correct. Recently, Chuang *et al.* proved [11][12] that a CIOQ switch using stable matching [19] algorithm and critical cells first (CCF) insertion policy with speed up equal to two can exactly mimic an output-queued (OQ) switch that uses push-in first-out (PIFO) queueing policy.

CHAPTER 3

THE EARLIEST DUE DATE FIRST MATCHING (EDDFM) ALGORITHM

Consider an $N \times N$ IQ switch consisting of N inputs, N outputs, and a non-blocking switch fabric. To provide QoS features, switch resources such as the bandwidth and buffers are allocated on a per-session basis. Let $I_{i,j,k}$ be the kth session in $Q_{i,j}$, the VOQ directed to output j at input i, with arrival rate $r_{i,j,k}$. Denote $A_{i,j}$ as the arrival process of VOQ $Q_{i,j}$. Then, the arrival rate of $A_{i,j}$ can be expressed as $r_{i,j} = \sum_k r_{i,j,k}$. An arrival process A_i, which is the aggregate arrival process to input i, is said to be uniform if $r_{i,j} = r_{i,j'}$, $\forall\ j \neq j'$, $0 \leq j, j' \leq N-1$. Otherwise, the process is said to be non-uniform. The traffic pattern is admissible if and only if

$$\sum_{j=0}^{N-1} r_{i,j} \leq 1, \forall i \tag{3.1}$$

and

$$\sum_{i=0}^{N-1} r_{i,j} \leq 1, \forall j. \tag{3.2}$$

3.1 The Algorithm

When a cell belonging to session $I_{i,j,k}$ in $Q_{i,j}$ arrives at timeslot n, the earliest due date first matching (EDDFM) algorithm sets the initial weight of the cell to $\mathcal{P}_{i,j,k}(n) = \mathcal{P}_m - \Gamma_{i,j,k}$, where $\Gamma_{i,j,k}$ is the delay bound of session $I_{i,j,k}$ and $\mathcal{P}_m$ is an integer that is greater than $max_{\forall i,j,k}(\Gamma_{i,j,k})$. Then, the cell is inserted in the queue at the position closest possible to the HOL so that all the cells beyond this cell have smaller weights. If a cell in the queue is served in a timeslot, it will be deleted from the queue; otherwise, its weight will increase by one. The weight of VOQ $Q_{i,j}$ at timeslot n, $w_{i,j}(n)$, is set to the weight of the HOL cell of this queue if $Q_{i,j}$ is not empty; otherwise $w_{i,j}(n) = 0$.

Let $\underline{W}_i(n) = (w_{i,1}(n), w_{i,2}(n), \ldots, w_{i,N}(n))^T$ be the weight vector of input i and $\underline{S}_i(n) = (S_{i,1}(n), S_{i,2}(n), \ldots, S_{i,N}(n))^T$ be the service vector associated with input i. The EDDFM performs the following for each timeslot n:

1. Each input i set the weight of every *VOQ* $Q_{i,j}$ to the weight of the HOL cell, and sends the weight vector $\underline{W}_i(n)$ to the scheduler.

2. The scheduler searches for a match that achieves the maximum aggregate weight under the constraint of unique pairing, i.e.,

$$\mathbf{S} = \arg\max_{\mathbf{S}}[\sum_{i,j} S_{i,j}(n) w_{i,j}(n)]$$

   such that $\sum_i S_{i,j}(n) = \sum_j S_{i,j}(n) = 1$, sends the service vector $\underline{S}_i(n)$ to the corresponding input, and uses the matching matrix $(S_{i,j}(n))$ to configure the fabric.

3. Each input selects the HOL cell from the matched VOQ indicated by $\underline{S}_i(n)$ for transmission.

Lemma 1 *Using EDDFM algorithm, the weights of the VOQs are stable for all admissible independent traffic, i.e.,* $E[w_{i,j}(n)] < \infty, \forall i, j, n$.

Proof: *The proof of this Lemma is similar to the proof of the stability of OCF in [42]. The weight of $Q_{i,j}$ at timeslot $n+1$, $w_{i,j}(n+1)$, is set as follows:*

$$w_{i,j}(n+1) = 0, \tag{3.3}$$

if $Q_{i,j}$ is empty at timeslot $n+1$.

$$w_{i,j}(n+1) = \mathcal{P}_{i,j,k}(n+1), \tag{3.4}$$

if a cell arrives at $Q_{i,j}$ and its weight is larger than the weight of the HOL cell.

$$w_{i,j}(n+1) = w_{i,j}(n) + 1, \tag{3.5}$$

if the queue is not served in this timeslot, and no cell with a larger weight arrives.

$$w_{i,j}(n+1) = w_{i,j}(n) - \tau_{i,j}(n) + 1, \tag{3.6}$$

otherwise, where $\tau_{i,j}(n)$ is the difference of the weights between HOL cell and the cell behind it at timeslot n, and $\tau_{i,j}(n) \geq 0$.

Set, $\tilde{w}_{i,j}(n+1)$, the approximate next-state weight of $Q_{i,j}$ to be:

$$\tilde{w}_{i,j}(n+1) = w_{i,j}(n) + 1 - S_{i,j}(n)\tau_{i,j}(n), \tag{3.7}$$

we can obtain:

$$w_{i,j}(n+1) \leq max[\tilde{w}_{i,j}(n+1), \mathcal{P}_m]. \tag{3.8}$$

Let

$$\underline{W}(n) = [w_{0,0}(n), ..., w_{0,N-1}(n), w_{1,0}(n), ..., w_{N-1,N-1}(n)],$$

$$\underline{\tilde{W}}(n+1) = [\tilde{w}_{0,0}(n+1), ..., \tilde{w}_{0,N-1}(n+1), \tilde{w}_{1,0}(n+1), ..., \tilde{w}_{N-1,N-1}(n+1)],$$

and

$$\mathbf{\Lambda} = diag\{r_{0,0}, ..., r_{0,N-1}, r_{1,0}, ..., r_{N-1,N-1}\}.$$

Define the quadratic Lyapunov function $\mathcal{L}(\cdot)$ as:

$$\mathcal{L}(\underline{W}(n)) = \underline{W}^T(n)\Lambda\underline{W}(n) \tag{3.9}$$

Then,

$$\begin{aligned}\mathcal{L}(\underline{\tilde{W}}(n+1)) &= \underline{\tilde{W}}^T(n+1)\Lambda\underline{\tilde{W}}(n+1)\\ &= (\underline{W}(n) + \underline{1} - [\underline{S}\cdot\underline{\tau}(n)])^T\Lambda(\underline{W}(n) + \underline{1} - [\underline{S}\cdot\underline{\tau}(n)])\\ &= \underline{W}(n)^T\Lambda\underline{W}(n) + 2\underline{W}(n)^T\underline{r} - 2\underline{W}^T(n)[\underline{S}\cdot\underline{\tau}(n)\cdot\underline{r}] + \sum_{i,j} r_{i,j}\\ &\quad -2\sum_{i,j} S_{i,j}(n)\cdot\tau_{i,j}(n)\cdot r_{i,j} + \sum_{i,j} S^2_{i,j}(n)\cdot\tau^2_{i,j}(n)\cdot r_{i,j},\end{aligned}$$

where

$$\underline{S} = [S_{0,0}, ..., S_{0,N-1}, S_{1,0}, ..., S_{N-1,N-1}],$$

$$\underline{r} = [r_{0,0}, ..., r_{0,N-1}, r_{1,0}, ..., r_{N-1,N-1}],$$

$$\underline{\tau}(n) = [\tau_{0,0}(n), ..., \tau_{0,N-1}(n), \tau_{1,0}(n), ..., \tau_{N-1,N-1}(n)],$$

$$[\underline{S} \cdot \underline{\tau}(n)] = [S_{0,0} \cdot \tau_{0,0}(n), ..., S_{N-1,N-1} \cdot \tau_{N-1,N-1}(n)],$$

and

$$[\underline{S} \cdot \underline{\tau}(n) \cdot \underline{r}] = [S_{0,0} \cdot \tau_{0,0}(n) \cdot r_{0,0}, ..., S_{N-1,N-1} \cdot \tau_{N-1,N-1}(n) \cdot r_{N-1,N-1}].$$

Since $E[\tau_{i,j} r_{i,j}] = 1$, *we have:*

$$E[\mathcal{L}(\underline{\tilde{W}}(n+1)) - \mathcal{L}(\underline{W}(n)) \mid \underline{W}(n)]$$

$$= 2(\underline{W}^T(n)\underline{r} - \underline{W}^T(n)\underline{S}(n)) + \sum_{i,j} r_{i,j} - 2\sum_{i,j} S_{i,j}(n) + \sum_{i,j} \frac{S^2_{i,j}(n)}{r_{i,j}}$$

Since $\sum_{i,j} r_{i,j} \le N$, $\sum_{i,j} S_{i,j}(n) \ge 0$, *and* $\sum_{i,j} \frac{S^2_{i,j}(n)}{r_{i,j}} \le \mathcal{C}_1 < \infty$, *where* $\mathcal{C}_1$ *is a non-negative constant, we have*

$$E[\mathcal{L}(\underline{\tilde{W}}(n+1)) - \mathcal{L}(\underline{W}(n)) \mid \underline{W}(n)] \le \mathcal{C}_2 + 2(\underline{W}^T(n)\underline{(r)} - \underline{W}^T(n)\underline{S}(n)), \quad (3.10)$$

where $\mathcal{C}_2 = N + \mathcal{C}_1 > 0$.

Considering the second term of the right side of inequality Eq. 3.10, assume $\underline{r}_m$ *is a vector that satisfies* $||\underline{r}_m|| = N$ *and* $\underline{r} \le (1-\xi)\underline{r}_m$, *where* $0 < \xi < 1$. *Then,*

$$\underline{W}^T(n)\underline{r} - \underline{W}^T(n)\underline{S}(n) \le \underline{W}^T(n)(1-\xi)\underline{r}_m - \underline{W}^T(n)\underline{S}(n)$$

$$= \underline{W}^T(n)(\underline{r}_m - \underline{S}(n)) - \xi\underline{W}^T(n)\underline{r}_m \le -\xi\underline{W}^T(n)\underline{r}_m,$$

and

$$\underline{W}^T(n)\underline{r} - \underline{W}^T(n)\underline{S}(n) \le -\xi||\underline{W}(n)|| \cdot ||\underline{r}_m|| \cdot \cos\theta, \quad (3.11)$$

where θ *is the angle between* $\underline{W}(n)$ *and* $\underline{r}_m$. *Since* $W_{i,j}(n) > 0$ *if and only if* $r_{i,j} > 0$, *we know that* $cos\theta > 0$. *Specifically,*

$$\cos\theta = \frac{\underline{W}^T(n)\underline{r}}{||\underline{W}(n)|| \cdot ||\underline{r}||} \ge \frac{W_m(n)\gamma}{||\underline{W}(n)||\sqrt{N}},$$

where $W_m(n)$ is the maximum of all $w_{i,j}(n)$ and γ is the minimum of the non-negative $r_{i,j}, \forall i,j$. Thus,

$$\cos\theta \geq \frac{\gamma}{N\sqrt{N}}. \tag{3.12}$$

From Eq. 3.10, Eq. 3.11, and Eq. 3.12, we have

$$E[\mathcal{L}(\underline{\tilde{W}}(n+1)) - \mathcal{L}(\underline{W}(n)) \mid \underline{W}(n)]$$

$$\leq \mathcal{C}_2 - 2\xi\frac{\gamma}{N\sqrt{N}}||\underline{W}(n)||.$$

Let $\varepsilon = 2\xi\frac{\gamma}{N\sqrt{N}}$, which is larger than 0, we can obtain:

$$E[\mathcal{L}(\underline{\tilde{W}}(n+1)) - \mathcal{L}(\underline{W}(n)) \mid \underline{W}(n)] \leq \mathcal{C}_2 - \varepsilon||\underline{W}(n)||. \tag{3.13}$$

Since,

$$w_{i,j}(n+1) \leq max[\tilde{w}_{i,j}(n+1), \mathcal{P}_m],$$

we have

$$\underline{W}^T(n+1)\mathbf{\Lambda}\underline{W}(n+1) \leq \underline{\tilde{W}}^T(n+1)\mathbf{\Lambda}\underline{\tilde{W}}(n+1) + \sum_{i,j}\mathcal{P}_m^2 r_{i,j}. \tag{3.14}$$

Hence,

$$\underline{W}^T(n+1)\mathbf{\Lambda}\underline{W}(n+1) \leq \underline{\tilde{W}}^T(n+1)\mathbf{\Lambda}\underline{\tilde{W}}(n+1) + \mathcal{P}_m^2 N. \tag{3.15}$$

Note that $\mathcal{P}_m^2 N$ is a constant. Then,

$$E[\mathcal{L}(\underline{W}(n+1)) - \mathcal{L}(\underline{W}(n)) \mid \underline{W}(n)] \leq \mathcal{P}_m^2 N + \mathcal{C}_2 - \varepsilon||\underline{W}(n)||,$$

or

$$E[\mathcal{L}(\underline{W}(n+1)) - \mathcal{L}(\underline{W}(n)) \mid \underline{W}(n)] \leq \mathcal{C} - \varepsilon||\underline{W}(n)||, \tag{3.16}$$

where $\epsilon > 0$ and $\mathcal{C} = \mathcal{P}_m^2 N + \mathcal{C}_2$ is constant. Eq. (3.16) indicates that the weights of the VOQs are stable under the EDDFM algorithm, i.e., $E[||\underline{W}(n)||] < \infty, \forall n$

Denote $L_{i,j}(n)$ as the queue occupancy of $Q_{i,j}$ at timeslot n. A switch is stable if $E[L_{i,j}(n)] < \infty, \forall i,j,n$.

Theorem 1 *A switch using EDDFM algorithm is stable for all admissible independent traffic.*

Proof: *Under EDDFM algorithm, the queue occupancy is always less than or equal to the weight of HOL cell, so the queue occupancies are stable.*

Theorem 2 *Under EDDFM algorithm, no session will be starved.*

Proof: *A cell's weight will keep increasing until it is served. Thus, EDDFM is a starvation-free algorithm.*

3.2 Simulation Results

A 4×4 input-queued switch was considered for simulations in which the bursty traffic was generated based on the $on - off$ traffic model. The average burst length was chosen to be 20 cells and the burstiness was 2. The traffic was non-symmetric, i.e., the arrival rates of the VOQs in the same input were different and were 0.5, 1, 2, and 5Mbps. Two sessions in each VOQ, a fast session with a rate four times that of a slow session, were generated. A traffic load of 0.9 was assumed, and each simulation lasted through 100 seconds.

Three levels of delay bound, which are short, medium and long, were assumed in the simulation. The delay bounds are assigned according to the following rules: sessions with rates over 10% of the link capacity are treated as fast sessions and are assigned short delay, sessions with rates between 1% and 10% of the link capacity are treated as medium sessions and are assigned short and medium delay randomly, and sessions with rates less than 1% of the link capacity are treated as slow sessions and are assigned short, medium and long delay randomly. The medium delay and long delay are set to five times and ten times of the short delay, respectively. The configuration of delay bounds of each session remains the same for different algorithms for comparison purpose. The probabilities of cell overdue for different algorithms are shown in Fig. 3.1. The values of the delay bound in the figure are associated with short delay.

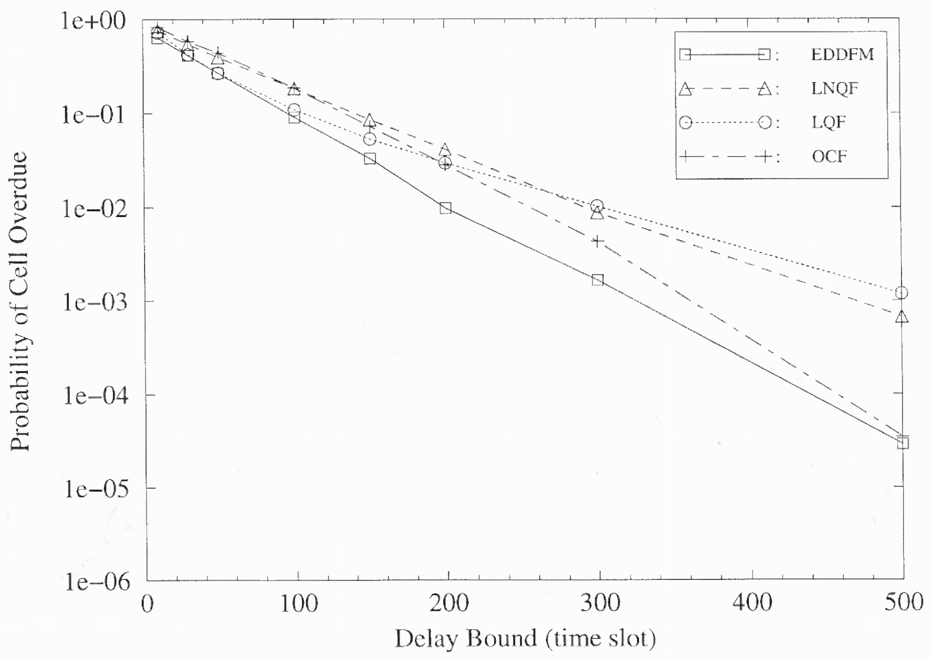


Figure 3.1 Comparison of probability of cell overdue under EDDFM and other existing algorithms

The fairness of a scheduler can be defined as [21]:

$$\mathcal{F} = \max_{\forall i,j,\ j\neq i} \left| \frac{T_i(t_1,t_2)}{r_i} - \frac{T_j(t_1,t_2)}{r_j} \right|,$$

where $T_i(t_1, t_2)$ is the number of cells delivered for session i during the time interval $[t_1, t_2]$, and r_i is the rate of session i. The fairness is the maximum difference of the normalized service time, which is the service a session received normalized by its rate, among all sessions. It provides a metric on how fair a server is. The smaller the amount of fairness, the fairer the server is. Table 3.1 summarizes the performance comparison among EDDMF, LNQF, LQF, and OCF, where $D_{i,j,k}$ is the average delay of session $I_{i,j,k}$. Table 3.1 and Fig. 3.1 show that EDDFM has the lower probability of cell overdue and better fairness than LNQF, LQF, and OCF.

Table 3.1 Statistics of the simulation results of EDDFM algorithm

schedulers	EDDFM	LNQF	LQF	OCF
$D_{0,0,1}$ (timeslot), $r_{0,0,1}$=0.1Mbps	89.0	22.5	535.6	62.7
$D_{0,0,2}$ (timeslot), $r_{0,0,2}$=0.4Mbps	75.4	40.8	296.6	63.8
$D_{0,1,1}$ (timeslot), $r_{0,1,1}$=0.2Mbps	62.4	38.7	193.7	66.8
$D_{0,1,2}$ (timeslot), $r_{0,1,2}$=0.8Mbps	44.7	59.7	85.1	67.4
$D_{0,2,1}$ (timeslot), $r_{0,2,1}$=0.4Mbps	73.5	49.4	114.7	64.8
$D_{0,2,2}$ (timeslot), $r_{0,2,2}$=1.6Mbps	51.9	64.4	44.6	68.2
$D_{0,3,1}$ (timeslot), $r_{0,3,1}$=1Mbps	47.7	52.1	76.0	57.9
$D_{0,3,2}$ (timeslot), $r_{0,3,2}$=4Mbps	47.4	62.4	24.7	59.5
average queue length (cell)	50.3	61.3	62.2	64.5
fairness	7.5	10.4	20.0	8.1
average transmission time (timeslot) per burst	97.0	125.9	130.7	97.6

Furthermore, EDDFM is proved analytically to be stable and starvation-free under all admissible independent traffic patterns.

CHAPTER 4

THE SHADOW DEPARTURE TIME ALGORITHM (SDTA) AND THE ITERATIVE SDTA (ISDTA)

Since MWM is used by LQF, OCF, and EDDFM, the complexity of these algorithms are very high. To facilitate the simplicity, GS-LQF and GS-OCF, the approximations of LQF and OCF, were introduced in [42] using stable marriage matching [19] which can be solved by Gale-Shapley Algorithm (GSA). In this chapter, the shadow departure time algorithm (SDTA) and iterative SDTA (ISDTA) are proposed in order to achieve better performance than GS-LQF and GS-OCF with the same level of complexity.

4.1 The Algorithms

It is assumed that there exists a shadow $N \times N$ FIFO OQ switch and the exactly same traffic going to the IQ switch is fed into the shadow switch concurrently. $SDT(\mathbf{c})$, the shadow departure time of a cell $\mathbf{c}$, is defined as the time that the cell departs the shadow switch. Since FIFOs are used both in the VOQs of the IQ switch and the output queues of the shadow switch, in the same VOQ the cell that arrives later will have a larger SDT and the HOL cell will have the smallest SDT among all the cells belonging to the same VOQ.

Using SDTA, the weight of a cell $\mathbf{c}$ is set to $w(\mathbf{c}) = SDT(\mathbf{c}) - n$, where n is the current time. $w_{i,j}$, the weight of $Q_{i,j}$, is defined as:

$$w_{i,j} = \begin{cases} w(\mathbf{c}^{\mathbf{0}}_{\mathbf{i,j}}(n)), & \text{if } Q_{i,j} \text{ is not empty,} \\ \infty, & \text{otherwise,} \end{cases}$$

in which $\mathbf{c}^{\mathbf{0}}_{\mathbf{i,j}}(n)$ is the HOL cell of $Q_{i,j}$ at time n. According to the above definition, the cell that has a smaller weight is more urgent to leave the switch. SDTA searches for a stable matching between inputs and outputs by setting the preference lists for every input and output following such a rule: input i prefers output j with smaller

$w_{i,j}$, and ties are broken randomly. Conversely, output j prefers input i with the smaller $w_{i,j}$, and ties are also broken randomly.

Iterative SDTA (ISDTA) is designed to find the maximal matching between inputs and outputs by iterations. It iterates the following three steps:

1. *Request:* Each unmatched input i sends a request to each output j if $Q_{i,j}$ is not empty.

2. *Grant:* If an unmatched output j receives any requests, it chooses the input i with the smallest weight $w_{i,j}$, and ties are broken randomly.

3. *Accept:* If an input i receives any grants, it chooses the output j with the smallest weight $w_{i,j}$, and ties are broken randomly.

The maximal matching can be reached in N iterations.

It can be proved that the VOQ, which has the smallest weight, will always be chosen to transmit the HOL cell. If a HOL cell of a VOQ is not served in a timeslot, its weight will decrease by one, and thus it will eventually become small enough to be served. Hence, SDTA and ISDTA are starvation-free algorithms.

4.2 Performance of ISDTA without Speedup

The performance of SDTA, ISDTA, and other existing algorithms were simulated in a 16x16 switch. 256 *i.i.d.* flows, each of which belongs to a different input-output pair, were created in the simulations. Two types of traffic models were taken into consideration in the simulations: Bernoulli traffic and bursty traffic generated based on the $on-off$ traffic model. In the $on-off$ traffic model, the average burst length was chosen to be 10 cells and the peak cell rate was set to be the link capacity. Each simulation lasted for 1 million timeslots. Simulation results indicate that the performance of SDTA and ISDTA are identical. This result is not surprising, because

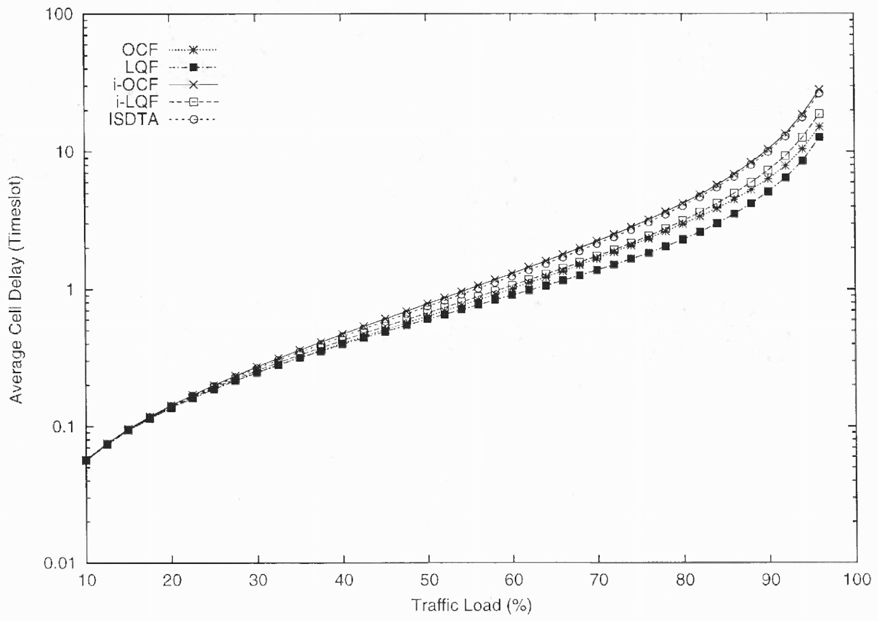


Figure 4.1 Average cell delay vs. traffic load under *i.i.d.* Bernoulli traffic.

it has already been demonstrated in [42], the performance of GS-OCF and GS-LQF which use GSA is identical to that of *i*-OCF and *i*-LQF which use iterative approach.

Figure 4.1 shows the average cell delay of OCF, LQF, *i*-OCF, *i*-LQF and ISDTA versus the traffic load under Bernoulli traffic. Fig. 4.2 shows the variance of cell delay versus the traffic load under the same traffic model. Figure 4.3 and 4.4 show the average cell delay and variance of cell delay versus the traffic load under $on - off$ traffic, respectively. Figure 4.1 – 4.4 indicate that ISDTA's average cell delay is slightly smaller than *i*-OCF and larger than *i*-LQF, and, on the other hand, ISDTA's variance of cell delay is slightly larger than that of *i*-OCF and smaller than *i*-LQF. LQF has the smallest average cell delay and OCF has the smallest variance of cell delay.

The performance of ISDTA and *i*-OCF are quite similar in terms of average cell delay and variance of cell delay. Yet, the difference of the two algorithms is found to be more profound after examining their cell delay distributions. Figure 4.5 shows the

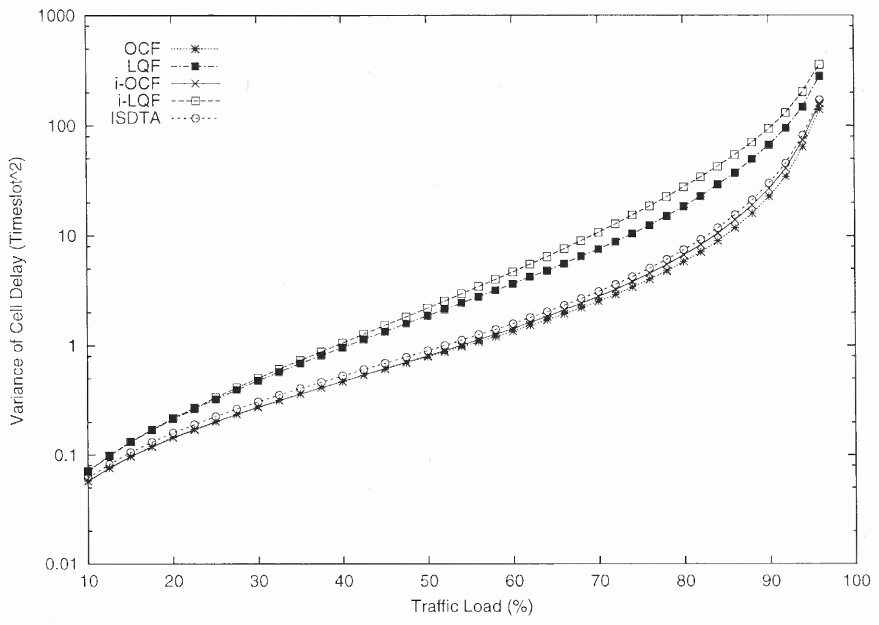


Figure 4.2 Variance of cell delay vs. traffic load under *i.i.d.* Bernoulli traffic.

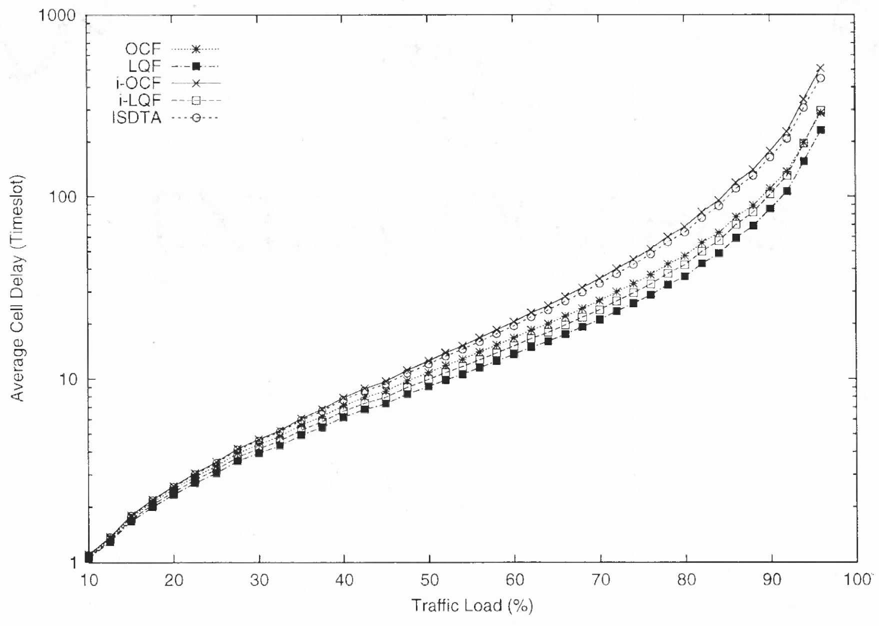


Figure 4.3 Average cell delay vs. traffic load under *i.i.d.* $on - off$ traffic.

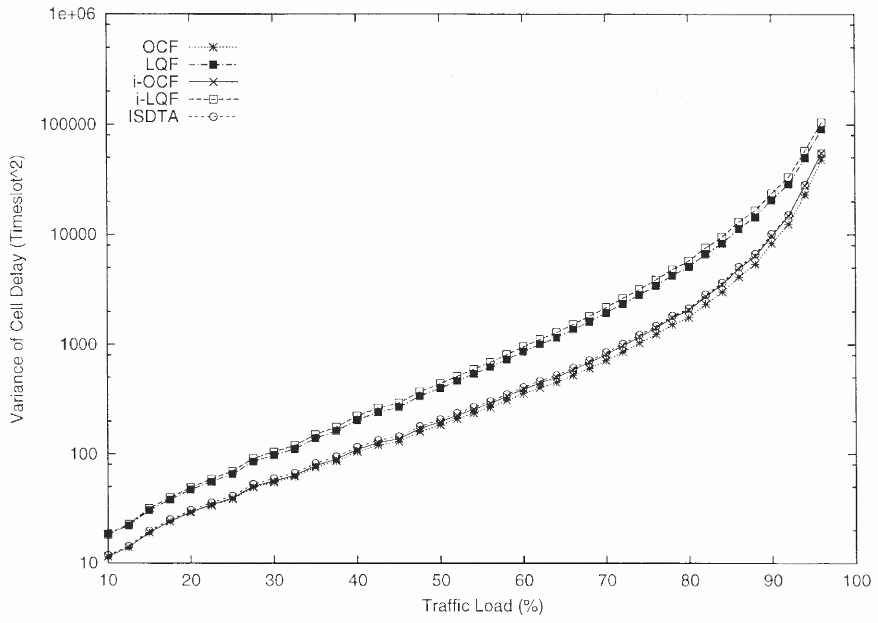


Figure 4.4 Variance of cell delay vs. traffic load under *i.i.d.* $on - off$ traffic.

distribution of the percentage of cells which experience various delays using *i*-OCF, *i*-LQF and ISDTA under *i.i.d.* Bernoulli traffic with a traffic load of 96%. Figure 4.7 shows the distribution under *i.i.d.* $on - off$ traffic with a traffic load of 80%. The curves in the above figures approximate the probability density functions of cell delay using the algorithms under the above conditions. Figure 4.6 and 4.8 plot the logarithm of the percentage versus cell delay in order to highlight the tails of the distributions. These figures show that *i*-LQF has a heavy tail, which shows that the number of cells which endure much longer latency is not desirable. Figure 4.8 also indicates that the tail of *i*-LQF diminishes much slower than that of *i*-OCF and ISDTA under bursty traffic. Thus, it is not difficult to understand why *i*-LQF has a smaller average cell delay but a larger variance of cell delay than *i*-OCF and ISDTA. In the real-time service, the cells with a delay larger than the maximum cell transfer delay (maxCTD) are considered to be lost [2]. Table 4.1 tabulates the cell loss ratio (CLR) of *i*-LQF, *i*-OCF, and ISDTA algorithms when maxCTD is 200, 300, and 400

Table 4.1 Cell loss ratio of i-LQF, i-OCF, and ISDTA algorithms

Algorithm	maxCTD = 200	maxCTD = 300	maxCTD = 400	maxCTD = 500
i-LQF	$4.37e-2$	$1.87e-2$	$8.44e-3$	$4.97e-3$
i-OCF	$1.42e-2$	$1.23e-3$	$1.33e-5$	$< 1.0e-7$
ISDTA	$1.52e-2$	$1.35e-3$	$1.20e-5$	$< 1.0e-7$

under 80% $on-off$ arrival. This result indicates that the CLR of i-LQF is so large that it may not be acceptable for the real-time service.

The tails corresponding to ISDTA and i-OCF are almost identical while the areas around their peaks are different. Integrating the curves in Fig. 4.5 and Fig. 4.7, the approximations of the cumulative distribution functions of cell delay can be obtained which are plotted in Fig. 4.9 and 4.10. This figure shows that the cumulative distributions of ISDTA are always larger than that of i-OCF under both conditions, which implies that for a given delay bound, more percentage of cells can be transmitted within the delay bound using ISDTA than using i-OCF. In other words, cells using ISDTA have a lower probability of overdue than those using i-OCF.

4.3 Performance of ISDTA with Speedup

Chuang *et al.* [12] proved that the necessity and sufficiency of speedup ω required for an $N \times N$ CIOQ switch to exactly mimic an $N \times N$ FIFO-OQ switch is $2 - 1/N$. In order to prove these theorems, they assume that some timeslots are normal, which have two scheduling phases; some timeslots are truncated, which have only one scheduling phase. There is one truncated timeslot out of every N timeslots, so the speedup is $2 - 1/N$. It means that such a speedup is the average, not the worst case. Since there are two scheduling phases in a normal timeslot, the instant speedup is two.

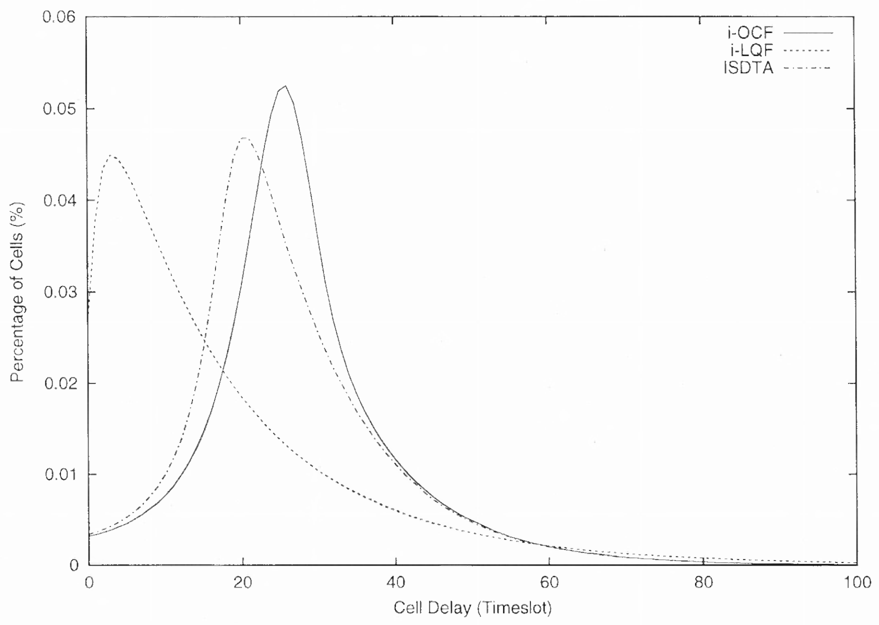


Figure 4.5 Cell delay distribution under *i.i.d.* Bernoulli traffic with traffic load of 96% in linear scale.

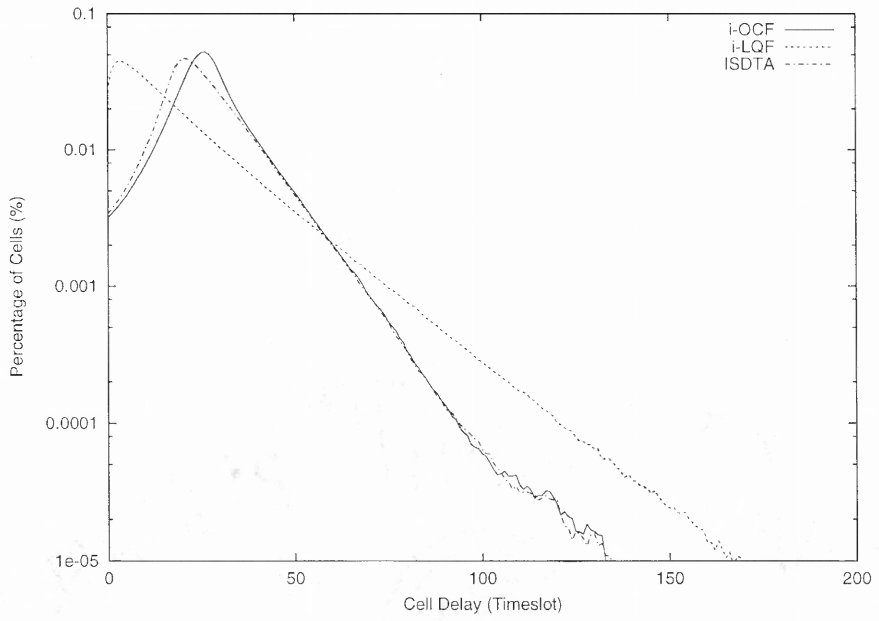


Figure 4.6 Cell delay distribution under *i.i.d.* Bernoulli traffic with traffic load of 96% in logarithmic scale.

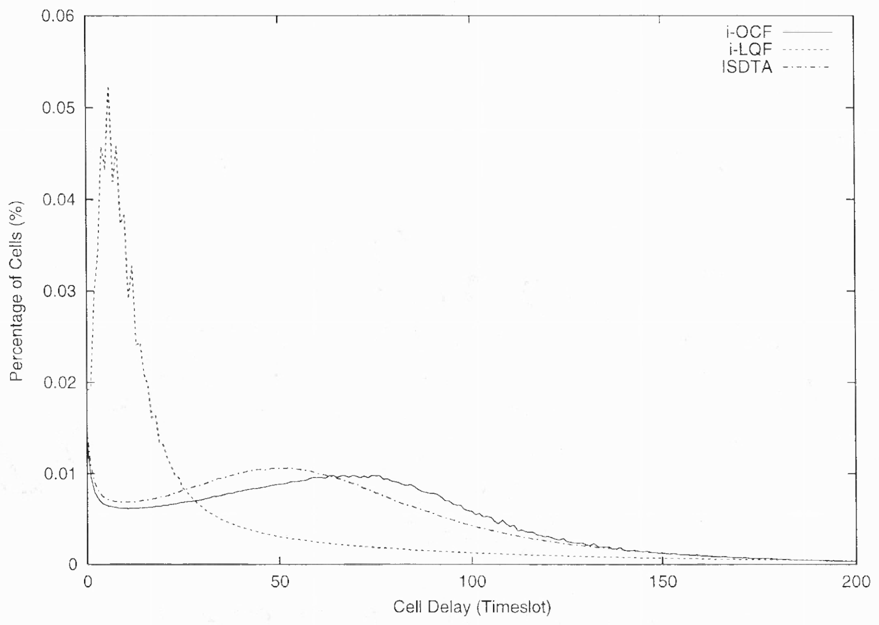


Figure 4.7 Cell delay distribution under *i.i.d.* $on - off$ traffic with traffic load of 80% in linear scale.

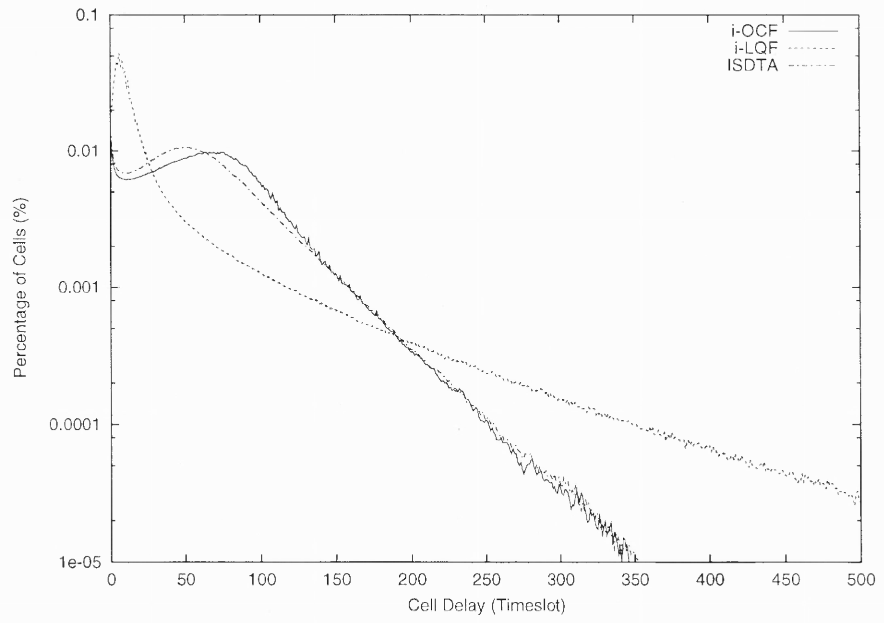


Figure 4.8 Cell delay distribution under *i.i.d.* $on - off$ traffic with traffic load of 80% in logarithmic scale.

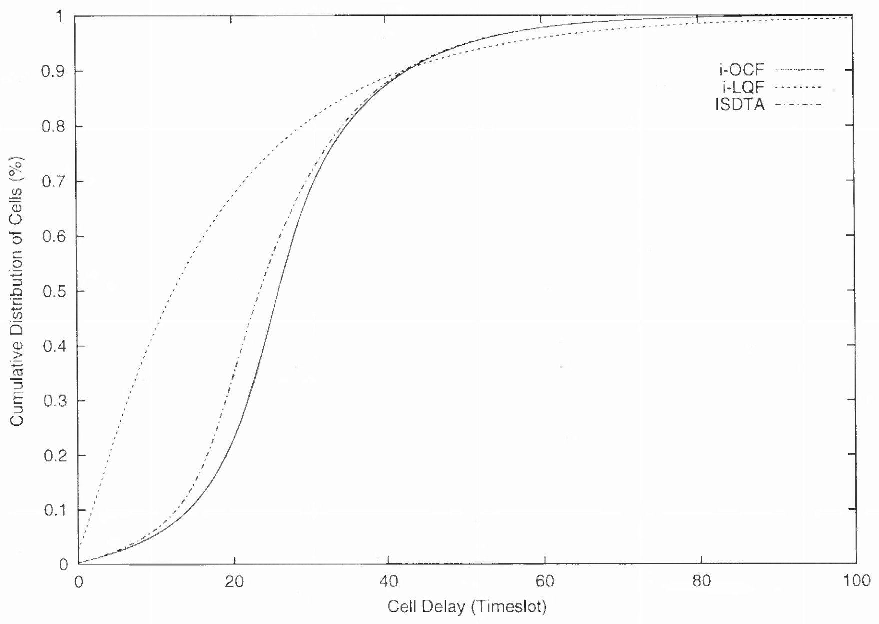


Figure 4.9 Cumulative distribution of cells under *i.i.d.* Bernoulli traffic with a load of 96%.

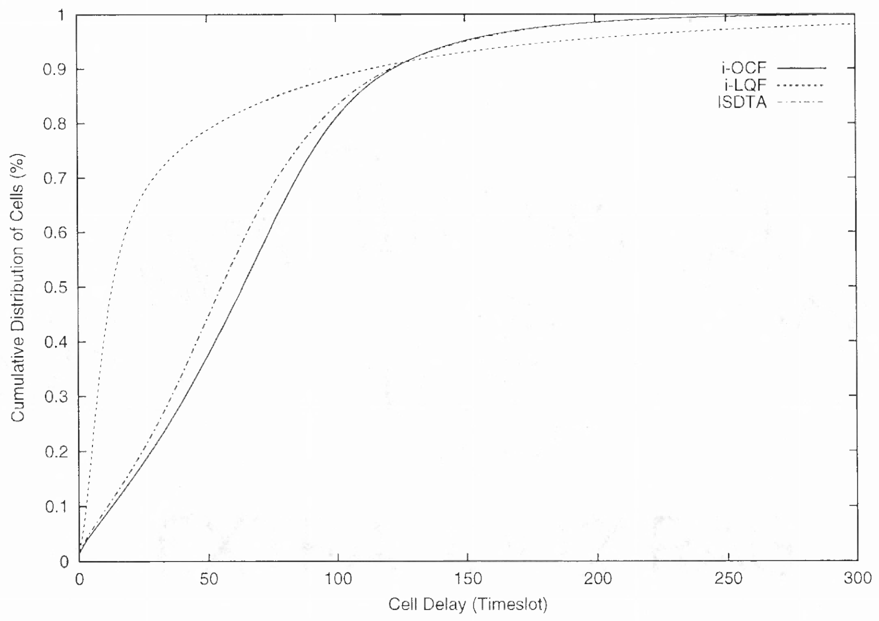


Figure 4.10 Cumulative distribution of cells under *i.i.d.* $on-off$ traffic with a load of 80%.

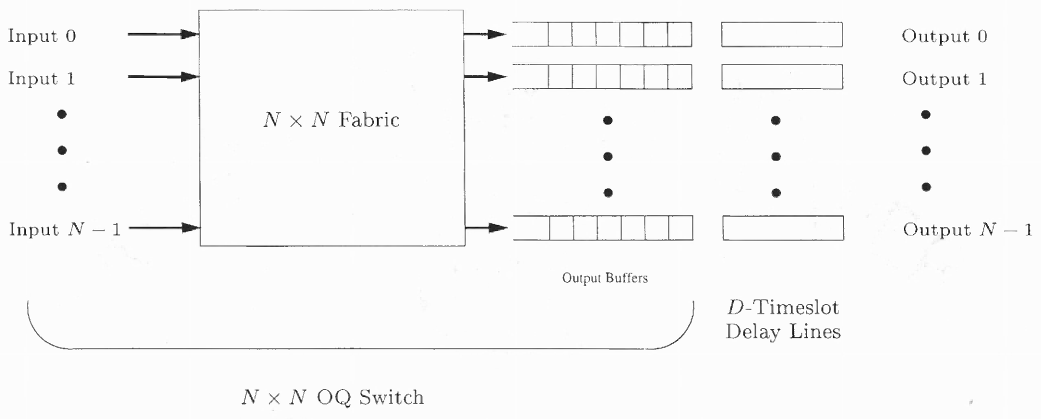


Figure 4.11 Delyed output queued switching architecture.

According to their analysis, it is impossible to exactly emulate an OQ switch using a CIOQ switch with a speedup less than $2 - 1/N$. Our question is whether it is possible to exactly emulate a delayed OQ switch with a less speedup. The architecture of a delayed OQ switch is shown in Fig. 4.11, which consists of an OQ switch and a D-timeslot delay line for each output. The performance of delayed OQ switch is identical to an OQ switch except that the cell delay over the former is exactly D timeslots larger than the later. Thus, some QoS features such as throughput and delay jitter will be identical for the two switches.

In the simulations, the exactly same traffic is fed into a CIOQ switch as well as a delayed FIFO-OQ switch with $D = F$. Define the overdue of a cell as the time that the cell leaves the CIOQ switch minus the time that the cell leaves the delayed OQ switch. The following figures plotted the number of cells versus their overdue using different algorithms and conditions. Figure 4.12(a) shows the result of the 16×16 CIOQ switch using i-OCF [42] algorithm when $F = 2$, $X = 3$, where $\omega = X/F = 1.50$. The traffic is $i.i.d.\ on-off$ traffic, with traffic load equal to 91% and $\beta = 0.1$. The workload is uniform. The simulation last for $100,000$ timeslots. Figure 4.12 shows that the maximum overdue is about 170 timeslots under these conditions.

Figure 4.13 shows the result using the algorithm described in [12] with CCF insertion policy when $F = 2$, $X = 2$, where $\omega = 1.50$. The traffic is *i.i.d.* $on - off$ traffic with the traffic load of 96% and $\beta = 0.02$. The simulation lasted for one million timeslots. The maximum overdue in this test is about 130 timeslots. Since this test has both a heavy traffic and a longer test time, the performance of this algorithm is better than i-OCF under these conditions.

Denote ISDTA-X/F as the CIOQ switch that uses ISDTA with parameters X and F. Figure 4.14 shows the distribution of cell delay overdue using ISDTA-3/2 under *i.i.d.* $on - off$ arrival with a traffic load of 96% and $\beta = 0.01$, which is more bursty than the traffic used for i-OCF and CCF. The maximum overdue of ISDTA-3/2 is only 1 timeslot. These figures demonstrate that the performance of ISDTA is much better than i-OCF and CCF in terms of cell delay overdue. Figure 4.15 compares the average cell delay of FIFO-OQ and ISDTA-3/2 versus the traffic load under *i.i.d.* $on - off$ traffic. Figure 4.15 indicates that the average delay of ISDTA-3/2 is very close to that of FIFO-OQ, implying that ISDTA-3/2 can achieve 100% throughput with low delay. Figure 4.16 shows the variance of cell delay versus the traffic load under the same traffic model. Note that the curves of the variance of cell delay of ISDTA-3/2 and FIFO-OQ are almost identical.

In this section, two new algorithms, SDTA and ISDTA, which have the identical performance, were proposed. ISDTA, which employs maximal matching, thus having a lower complexity than algorithms that uses MWM, can be used to improve upon existing maximal matching algorithms in terms of QoS features. The cell delay distribution using i-LQF exhibits a heavy tail especially under bursty traffic. This implies that more cells experience much longer latency than those using i-OCF and ISDTA. Simulations also show that ISDTA has a lower probability of overdue than those using i-OCF. It has been showed by simulations that the performance of a

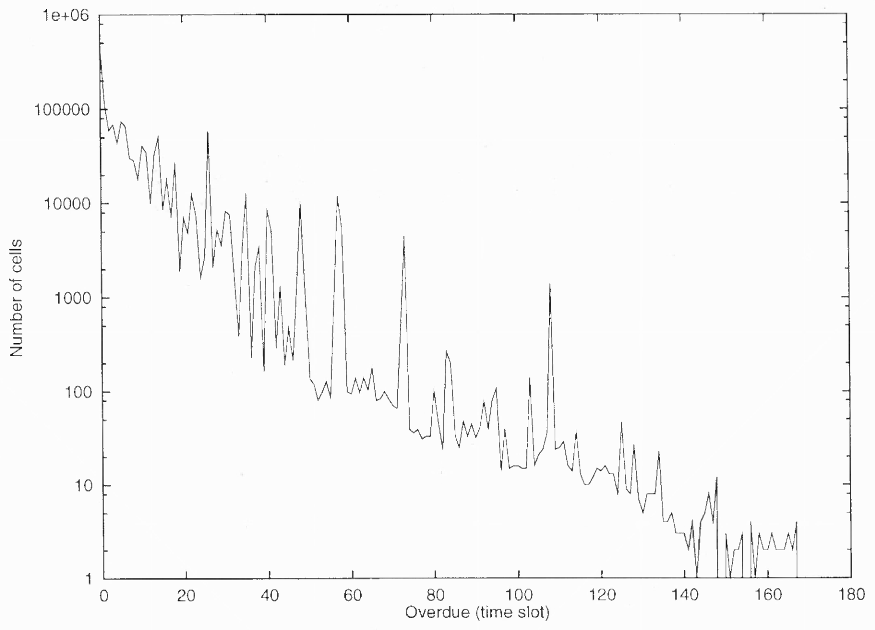


Figure 4.12 The distribution of cell delay overdue using i-OCF under $i.i.d.$ $on-off$ traffic with the speedup of 1.5.

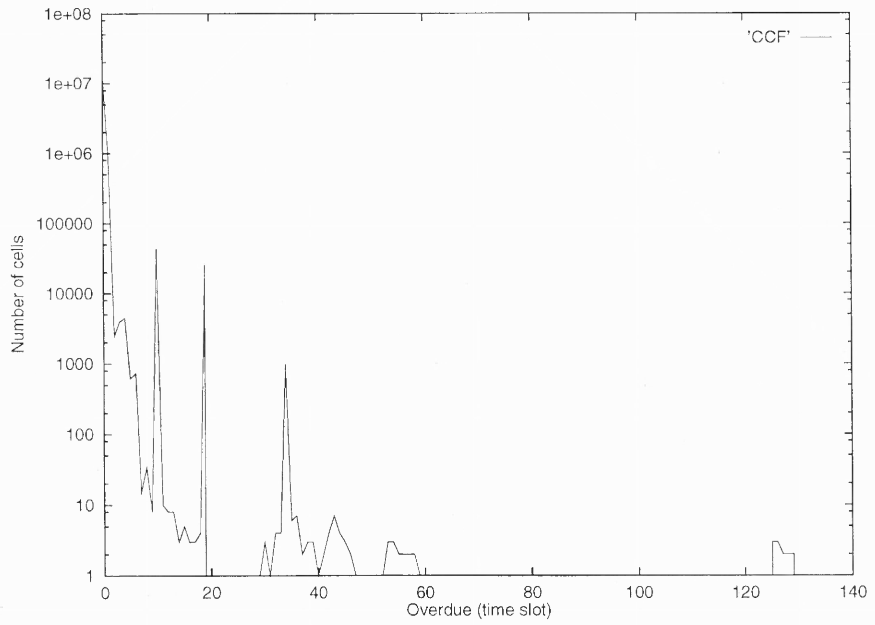


Figure 4.13 The distribution of cell delay overdue using CCF under $i.i.d.$ $on-off$ traffic with the speedup of 1.5.

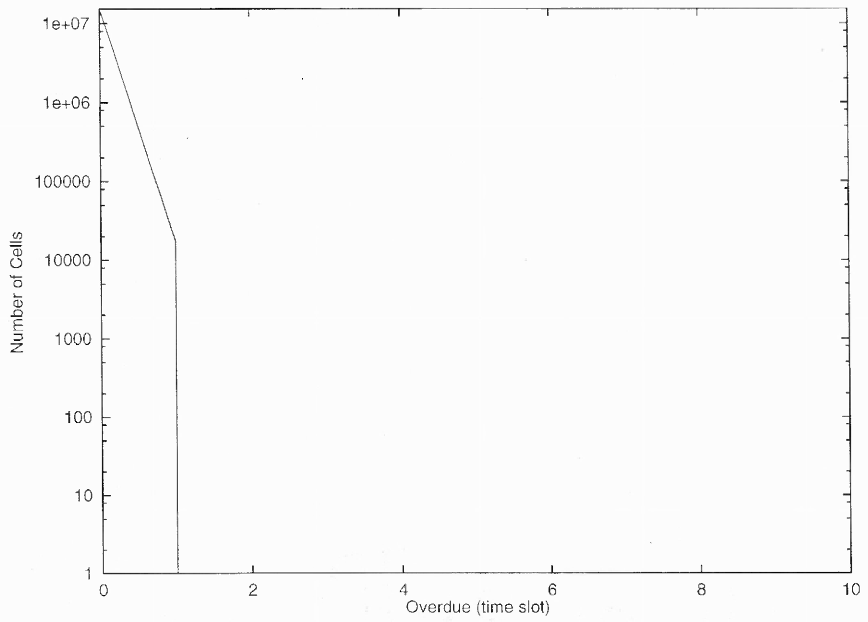


Figure 4.14 The distribution of cell delay overdue using ISDTA under *i.i.d. on−off* traffic with the speedup of 1.5.

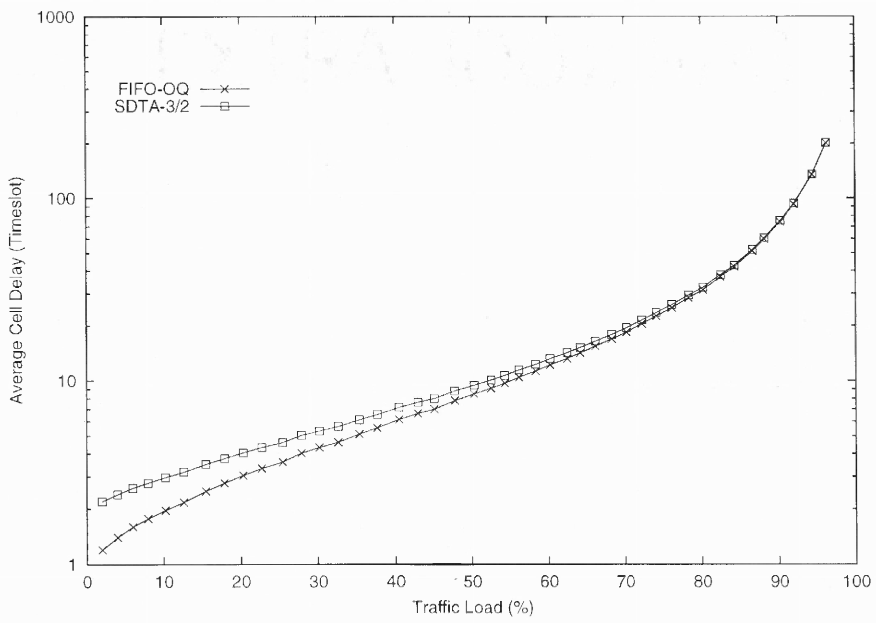


Figure 4.15 Average cell delay vs. traffic load under *i.i.d. on − off* arrival.

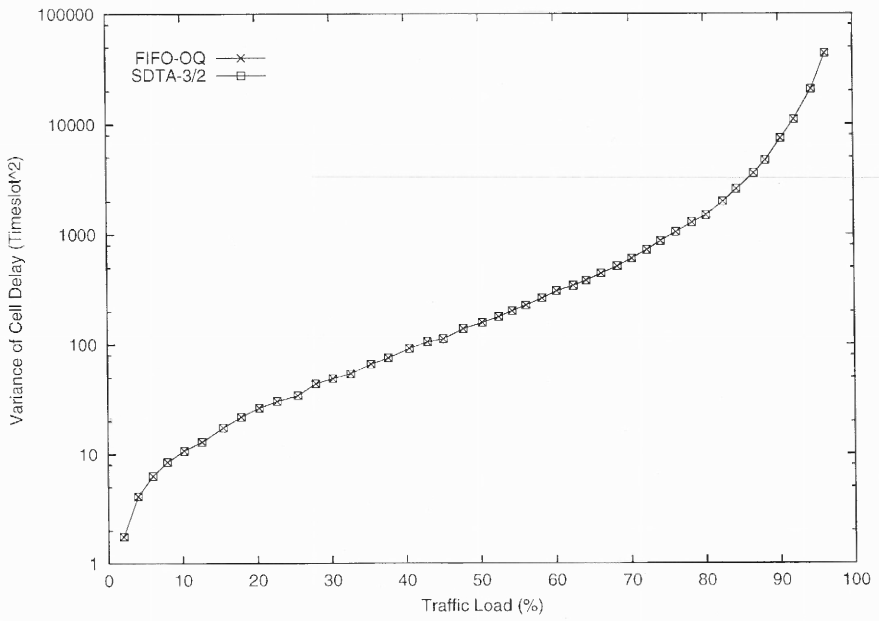


Figure 4.16 Variance cell delay vs. traffic load under $i.i.d.\ on - off$ arrival.

CIOQ switch using ISDTA with a speedup of 1.5 is similar to that of an OQ switch in terms of cell delay and throughput.

CHAPTER 5

THE ENHANCED BIRKHOFF-VON NEUMANN DECOMPOSITION (EBVND) ALGORITHM

Denote $r_{i,j}$ as the arrival rate of VOQ $Q_{i,j}$. The input traffic is said to be admissible if the following inequalities are satisfied:

$$\sum_{j=0}^{N-1} r_{i,j} \leq 1, \forall i, \tag{5.1}$$

$$\sum_{i=0}^{N-1} r_{i,j} \leq 1, \forall j. \tag{5.2}$$

The matrix $\mathbf{R} = (r_{i,j})$ satisfying Eq. (5.1) and (5.2) is said to be doubly substochastic. For any doubly substochastic matrix $\mathbf{R}$, there exists [6] a doubly stochastic matrix $\tilde{\mathbf{R}} = (\tilde{r}_{i,j})$ such that $r_{i,j} \leq \tilde{r}_{i,j}, \forall i, j$. Matrix $\tilde{\mathbf{R}}$ is said to be a doubly stochastic if

$$\sum_{j=0}^{N-1} \tilde{r}_{i,j} = 1, \forall i, \tag{5.3}$$

and

$$\sum_{i=0}^{N-1} \tilde{r}_{i,j} = 1, \forall j. \tag{5.4}$$

An algorithm to construct doubly stochastic matrix $\tilde{\mathbf{R}}$ from doubly substochastic matrix $\mathbf{R}$ is also provided in [6] with a computational complexity of $O(N^3)$. In this dissertation, a new algorithm, the weighted rate filling algorithm (WRFA), is introduced to perform this task:

Weighted Rate Filling Algorithm (WRFA)

1. *Define* $p_i = 1 - \sum_{j=0}^{N-1} r_{i,j}$. *Calculate* p_i *for all* i.

2. *Define* $q_j = 1 - \sum_{i=0}^{N-1} r_{i,j}$. *Calculate* q_j *for all* j.

3. *Calculate* $\Delta = N - \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} r_{i,j}$.

4. *Let* $\tilde{r}_{i,j} = r_{i,j} + \frac{p_i q_j}{\Delta}$.

Theorem 3 *Matrix $\tilde{\mathbf{R}} = (\tilde{r}_{i,j})$ constructed from doubly substochastic matrix $\mathbf{R} = (r_{i,j})$ is a doubly stochastic matrix.*

Proof: *Since $\mathbf{R} = (r_{i,j})$ is a doubly substochastic matrix, we have $p_i \geq 0, \forall i$, $q_j \geq 0, \forall j$, and $\Delta \geq 0$. Thus, $\tilde{r}_{i,j} = r_{i,j} + \frac{p_i q_j}{\Delta} \geq 0, \forall i, j$. For any j, we have*

$$\sum_{i=0}^{N-1} \tilde{r}_{i,j} = \sum_{i=0}^{N-1} r_{i,j} + \sum_{i=0}^{N-1} (\frac{p_i q_j}{\Delta}) = 1 - q_j + \frac{\sum_i (1 - \sum_j r_{i,j}) q_j}{\Delta}$$

$$= 1 - q_j + \frac{(N - \sum_{i,j} r_{i,j}) q_j}{\Delta} = \frac{\Delta - q_i \Delta + N q_i - q_i (N - \Delta)}{\Delta} = 1.$$

For any i, we also have $\sum_{j=0}^{N-1} \tilde{r}_{i,j} = 1$. Thus, matrix $\tilde{\mathbf{R}} = (\tilde{r}_{i,j})$ is a doubly stochastic matrix.

Compared to the original algorithm, WRFA is simpler and fairer. For example, when $\mathbf{R} = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}$, the original algorithm constructs the doubly stochastic matrix $\tilde{\mathbf{R}} = \begin{bmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.7 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.7 \end{bmatrix}$. Using WRFA, we get $\tilde{\mathbf{R}} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$. Apparently, WRFA shares the unreserved rate among the VOQs more fairly. The complexity of WRFA is $O(N^2)$ which is smaller than the original one.

5.1 The Birkhoff-von Neumann Decomposition (BVND) Algorithm

A doubly stochastic matrix $\tilde{\mathbf{R}}$ can be expressed as the linear combination of permutation matrices [6], $\tilde{\mathbf{R}} = \sum_k \phi_k \mathbf{P}_k$, where $\mathbf{P}_k$ is a permutation matrix, and $0 < \phi_k \leq 1$ such that $\sum_k \phi_k = 1$. The Birkhoff-von Neumann decomposition (BVND) algorithm schedules the cells by setting the connection of the crossbar according to the permutation matrix $\mathbf{P}_k$ with probability ϕ_k [6]. Let $C_{i,j}(n)$ be the cumulative number of timeslots for transmission that are assigned to $Q_{i,j}$ by timeslot n. Denote $A_{i,j}(n)$ as the total number of cells arrived in $Q_{i,j}$ at the end of timeslot n. Then, BVND can guarantee

$$C_{i,j}(m) - C_{i,j}(n) \geq (m-n)r_{i,j} - u_{i,j}, \tag{5.5}$$

for all i, j, $m > n$, where $u_{i,j} \leq U = N^2 - 2N + 2$, if Eq. (5.1) and (5.2) are satisfied [6]. Eq. (5.5) implies that if $A_{i,j}(n)$ conforms to $(\sigma_{i,j}, r_{i,j})$, i.e.,

$$A_{i,j}(m) - A_{i,j}(n) \leq (m-n)r_{i,j} + \sigma_{i,j}, \tag{5.6}$$

then the cell delay from input i to output j is bounded by $\lceil(\sigma_{i,j} + u_{i,j})/r_{i,j}\rceil$ using BVND [6]. The off-line and on-line computational complexity of this algorithm is $O(N^{4.5})$ and $O(logN)$, respectively [6].

5.2 The Enhanced Birkhoff-von Neumann Decomposition (EBVND) Algorithm

It is observed that BVND decomposition algorithm is not efficient enough because in certain timeslot it sets the crossbar connections solely according to the permutation matrix which is obtained from $\tilde{\mathbf{R}}$, and pays no attention to the current occupancy of VOQs. Thus, it is not surprising to see that the average cell delay of BVND is much larger than that of other algorithms such as OCF. For example, the connection requests at the current timeslot is shown in Fig 1.8. It is redrawn in Fig. 5.1(a), where the non-empty VOQs are filled with a cross. Suppose the permutation matrix selected by BVND at the current timeslot is

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{5.7}$$

The non-zero elements of $\mathbf{P}$ are represented by circles in Fig. 5.1(a). Among all the VOQs selected by $\mathbf{P}$, only VOQ $Q_{0,1}$ is non-empty which is shown by thick border box in Fig. 5.1(b). Hence, only one cell can be scheduled by BVND in the current

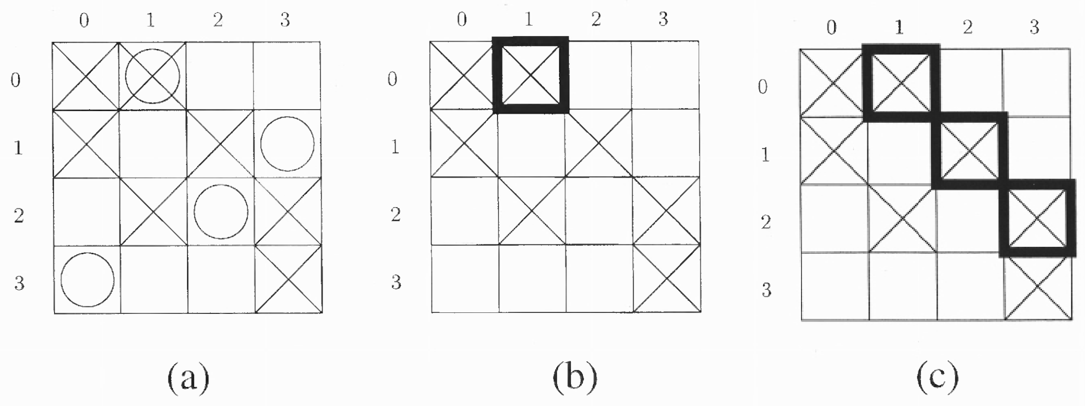


Figure 5.1 (a) Request graph and the selected permutation matrix. (b) One cell scheduled by BVND. (c) Two additional cells can be scheduled.

timeslot. However, two more VOQs, $Q_{1,2}$ and $Q_{2,3}$, can actually send cells across the fabric in the current timeslot without removing the cells scheduled by BVND as shown in Fig. 5.1(c).

Based on the above observation, the enhanced Birkhoff-von Neumann decomposition (EBVND) algorithm matches the inputs and outputs which are not matched by BVND, and attempts to make a maximal match in every timeslot. Many algorithms such as PIM and WWFA can be used to fill the "holes" left by BVND. Thus, EBVND will have a higher on-line computational complexity than BVND, with the expectation of having better performance.

Suppose IQ switch $\mathcal{B}$ uses BVND while IQ switch $\mathcal{E}$ uses EBVND. Denote ${Q_{i,j}}^{\mathcal{B}}$ and $Q^{\mathcal{E}}_{i,j}$ as the $Q_{i,j}$ of switch $\mathcal{B}$ and $\mathcal{E}$, respectively. Let $T^{\mathcal{B}}_{i,j}(n)$ be the cumulative number of cells dequeued from $Q^{\mathcal{B}}_{i,j}$ by the end of timeslot n, and $T^{\mathcal{E}}_{i,j}(n)$ be that of $Q^{\mathcal{E}}_{i,j}$. Define $T_{i,j}(n,m) = T_{i,j}(m) - T_{i,j}(n)$.

Lemma 2 *For any integer $m > n$, if both $Q^{\mathcal{B}}_{i,j}$ and $Q^{\mathcal{E}}_{i,j}$ are constantly backlogged from timeslot $n+1$ to timeslot m, then $T^{\mathcal{E}}_{i,j}(n,m) \geq T^{\mathcal{B}}_{i,j}(n,m)$.*

Proof: *When both $Q^{\mathcal{B}}_{i,j}$ and $Q^{\mathcal{E}}_{i,j}$ are not empty at certain timeslot, if a cell is dequeued from $Q^{\mathcal{B}}_{i,j}$, then a cell must be dequeued from $Q^{\mathcal{E}}_{i,j}$ according to the definition of EBVND.*

Let $L^{\mathcal{E}}_{i,j}(n)$ and $L^{\mathcal{B}}_{i,j}(n)$ be the length of $Q^{\mathcal{E}}_{i,j}$ and $Q^{\mathcal{B}}_{i,j}$ by the end of timeslot n, respectively.

Theorem 4 *If the exactly same traffic is fed into switch $\mathcal{B}$ and $\mathcal{E}$ concurrently and no cell is dropped, then $L^{\mathcal{E}}_{i,j}(n) \le L^{\mathcal{B}}_{i,j}(n)$ for any i, j, and n.*

Proof: *Consider any i, j, and n, if $L^{\mathcal{E}}_{i,j}(n) = 0$, then the theorem is proved, because $L^{\mathcal{B}}_{i,j}(n) \ge 0$. If $L^{\mathcal{E}}_{i,j}(n) > 0$, let $n_0 < n$ be the largest number such that $L^{\mathcal{E}}_{i,j}(n_0) = 0$. That is, from timeslot $n_0 + 1$ to timeslot n, $Q^{\mathcal{E}}_{i,j}$ is constantly backlogged. Denote $T^{\mathcal{B}}_{i,j}(n_0, n)_{backlog}$ be the number of cells dequeued from $Q^{\mathcal{B}}_{i,j}$ between timeslot $n_0 + 1$ and n inclusively when $Q^{\mathcal{B}}_{i,j}$ is constantly backlogged during $(n_0, n]$. From Lemma 2, we know that $T^{\mathcal{E}}_{i,j}(n_0, n) \ge T^{\mathcal{B}}_{i,j}(n_0, n)_{backlog}$. Thus, we have $T^{\mathcal{E}}_{i,j}(n_0, n) \ge T^{\mathcal{B}}_{i,j}(n_0, n)$, because $T^{\mathcal{B}}_{i,j}(n_0, n)_{backlog} \ge T^{\mathcal{B}}_{i,j}(n_0, n)$. Since $L^{\mathcal{E}}_{i,j}(n_0) = 0$, $L^{\mathcal{E}}_{i,j}(n_0) \le L^{\mathcal{B}}_{i,j}(n_0)$. From timeslot $n_0 + 1$ to n, the same number of cells are enqueued into $Q^{\mathcal{B}}_{i,j}$ and $Q^{\mathcal{E}}_{i,j}$, but more or the same number of cells are dequeued from $Q^{\mathcal{E}}_{i,j}$. Thus, $L^{\mathcal{E}}_{i,j}(n) \le L^{\mathcal{B}}_{i,j}(n)$ for any i, j, and n.*

Denote $D^{\mathcal{B}}_{\mathbf{c}}$ as the delay of certain cell $\mathbf{c}$ in switch $\mathcal{B}$, and $D^{\mathcal{E}}_{\mathbf{c}}$ as the delay in switch $\mathcal{E}$.

Theorem 5 *Assume all the VOQs in switch $\mathcal{B}$ and $\mathcal{E}$ are FIFOs. If the exactly same traffic is fed into switch $\mathcal{B}$ and $\mathcal{E}$ concurrently and no cell is dropped, then $D^{\mathcal{E}}_{\mathbf{c}} \le D^{\mathcal{B}}_{\mathbf{c}}$ for any cell $\mathbf{c}$.*

Proof: *If a cell $\mathbf{c}$ which is directed to output j arrives in input i at timeslot n, then both $Q^{\mathcal{B}}_{i,j}$ and $Q^{\mathcal{E}}_{i,j}$ will be backlogged until $\mathbf{c}$ is scheduled. At timeslot n, $L^{\mathcal{B}}_{i,j}(n) \ge L^{\mathcal{E}}_{i,j}(n)$. Assume cell $\mathbf{c}$ departs switch $\mathcal{E}$ at timeslot n_1, then $T^{\mathcal{E}}_{i,j}(n, n_1) \ge T^{\mathcal{B}}_{i,j}(n, n_1)$.*

Because the arrival of $\mathcal{B}$ and $\mathcal{E}$ are identical, the cell $\mathbf{c}$ *cannot leave switch* $\mathcal{B}$ *before timeslot* n_1*, if all the VOQs are FIFOs. So* $D_{\mathbf{c}}^{\mathcal{E}} \leq D_{\mathbf{c}}^{\mathcal{B}}$ *for any cell* $\mathbf{c}$.

Theorems 4 and 5 imply that the performance of EBVND is better than BVND in terms of throughput and cell delay guarantees. Thus, EBVND does provision QoS guarantees, since BVND was proven to provision QoS guarantees [6].

5.3 The Wave Front Birkhoff-von Neumann Decomposition (WFBVND) Algorithm

Wave front Birkhoff-von Neumann decomposition (WFBVND) algorithm, which is a special case of EBVND, finds more pairs in a match using a method similar to WWFA [65]. WFBVND divides a timeslot into N phases. Assume $\mathbf{P}$ is the permutation matrix selected by BVND. In the lth phase, WFBVND calculates matrix $\mathbf{V}_l = (V_{l,i,j})$, where $V_{l,i,j} = P_{(i+l)modN,j}$. For example, if $\mathbf{P}$ is defined by Eq. (5.7), then

$$\mathbf{V}_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \mathbf{V}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$\mathbf{V}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{V}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

During the lth phase, WFBVND checks the VOQs corresponding to the non-zero elements of $\mathbf{V}_l$, and adds the non-empty VOQs in the match if both its input and output are unmatched. Figure 5.2 shows an example, where the VOQs filled with crosses indicate the non-empty VOQs, the VOQs filled with circles indicate that the corresponding elements of $\mathbf{V}_l$ is 1, and the VOQs with thick border indicate they are scheduled to transmit cells. The on-line computational complex of WFBVND is $O(N^2)$.

The complexity of $O(N^2)$ may be costly for high-speed implementation. WFBVND with $logN$ iterations (WFBVND-$logN$) is thus introduced as a simplified version of WFBVND in order to reduce the complexity. It differs from WFBVND

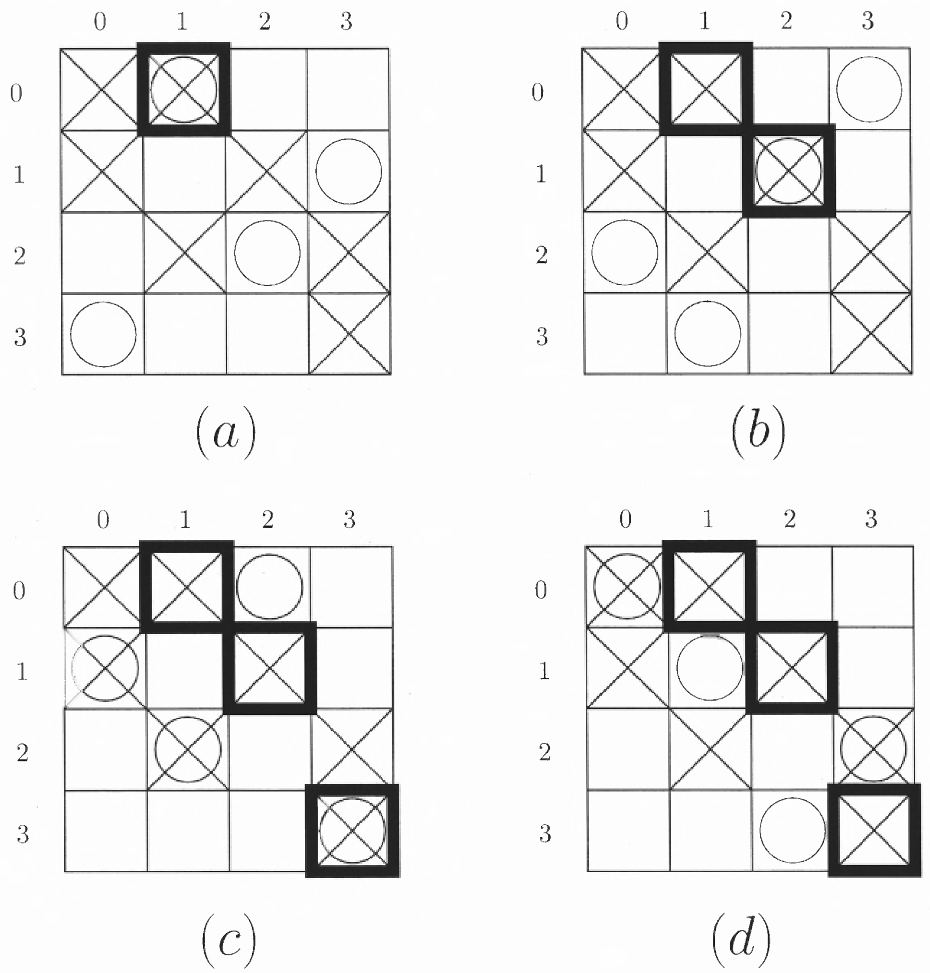


Figure 5.2 WFBVND algorithm: phase (a) 0, (b) 1, (c) 2, and (d) 3.

by only having the first $logN$ phases of WFBVND. Since WFBVND-$logN$ runs less phases than WFBVND, its performance is expected to be worse than WFBVND, but still better than BVND since it is the special case of EBVND. The on-line computational complexity of WFBVND-$logN$ is $O(NlogN)$.

5.4 Results and Discussion

The performance of the new and some other existing algorithms was simulated in a 16×16 IQ switch. 256 *i.i.d.* flows, each belonging to a different input-output pair, were created in the simulations. Traffic $A_{i,j}$ conforms to $(\sigma_{i,j}, r_{i,j})$ for all i, j, where $\sigma_{i,j}$ is set to be $1000r_{i,j}$. Thus, the delay bound is $1000 + \lceil u_{i,j}/r_{i,j} \rceil$ timeslots for BVND, WFBVND and WFBVND-$logN$.

Figure 5.3 shows the distribution of the percentage of cells which experience various delays over the switch using BVND, OCF, and WFBVND algorithm under the given traffic with a total traffic load of 84%. It shows clearly that the cell delay of BVND is much larger than OCF, while WFBVND is quite close to OCF. Figure 5.4 shows the average cell delay of BVND, WFBVND, WFBVND-$logN$, and OCF versus the traffic load under *i.i.d.* (σ, r) traffic. Figure 5.5 shows the variance of cell delay versus the traffic load under the same traffic model. Figure 5.4 and 5.5 indicate that the average cell delay and variance of WFBVND is much smaller than that of BVND and close to OCF. With a reduced complexity, the average delay and the delay variance of WFBVND-$logN$ are also significantly smaller than those of BVND.

The algorithms are also tested under unbalanced traffic load. In the 256 flows, half of them, which are selected randomly, are set to be inactive, and the active flows are assigned rates randomly under the admissible condition. Table 5.1 shows the average cell delay of all the flows from input 0 to the outputs using BVND, WFBVND, and WFBVND-$logN$. Table 5.2 tabulates the weighted average cell delay,

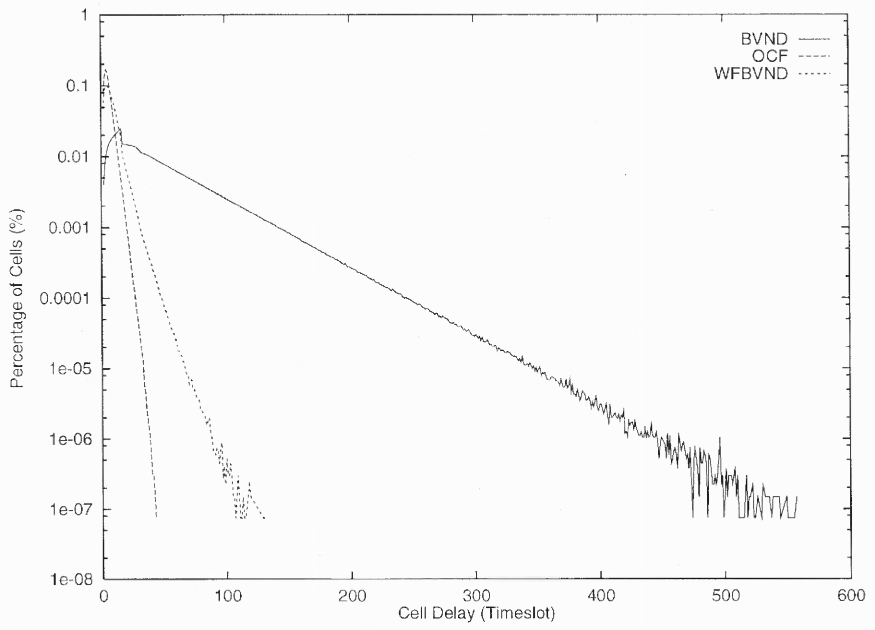


Figure 5.3 Comparison of cell delay distribution under 84% traffic load using BVND, OCF, and WFBVND.

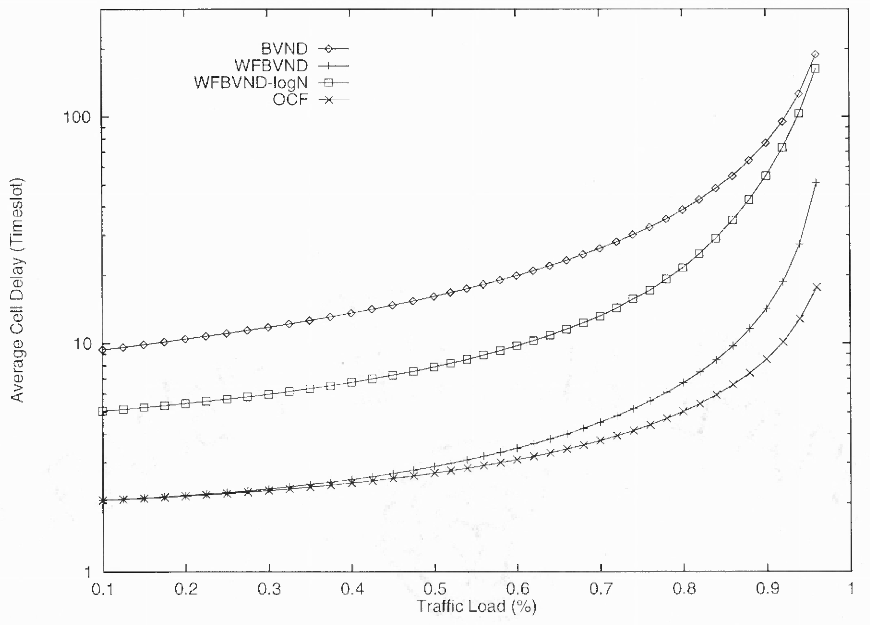


Figure 5.4 Average cell delay vs. traffic load under *i.i.d.* (σ, r) arrival.

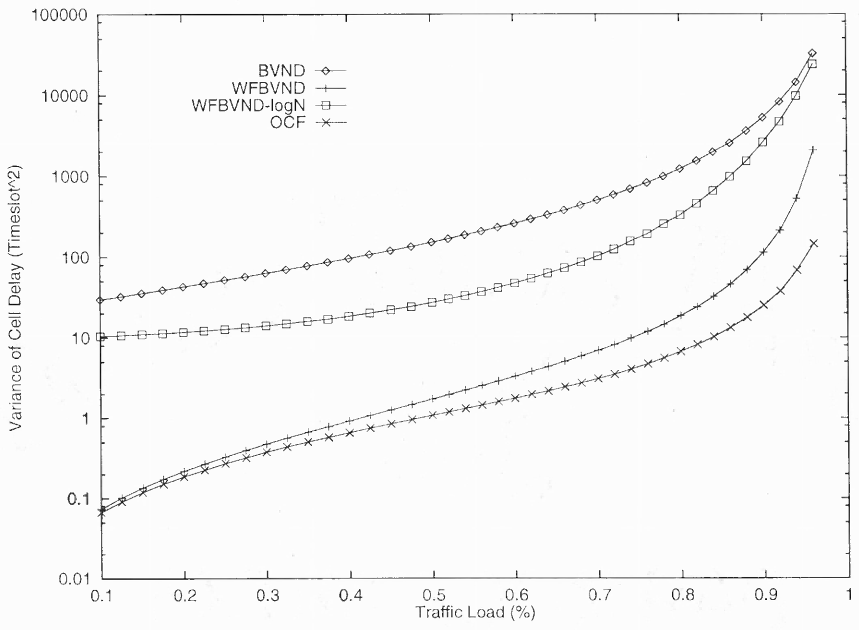


Figure 5.5 Variance of cell delay vs. traffic load under *i.i.d.* (σ, r) arrival.

Table 5.1 Average cell delay of the active flows in input 0

Algorithm	$D_{0,0}$	$D_{0,2}$	$D_{0,4}$	$D_{0,11}$	$D_{0,12}$	$D_{0,13}$	$D_{0,14}$	$D_{0,15}$
BVND	9.3	975	968	984	916	919	983	969.0
WFBVND-*logN*	8.6	4.6	11.2	984	905	552	512	138.0
WFBVND	6.6	2.9	4. 0	3.5	3.4	5.1	25	107.7

variance of cell delay, and average queue length of all the flows The results demonstrate that our proposed algorithms have also achieved much smaller average cell delay, cell delay variance, and average queue length in this traffic load.

A new class of VOQ scheduling algorithms with rate and delay guarantees has been proposed. Specifically, the performance of WFBVND and WFBVND-*logN* is compared with OCF and BVND. It has been demonstrated both by simulations and theoretical analysis that the new algorithms can achieve much smaller average cell delay and delay variance as well as provide QoS guarantees.

Table 5.2 Performance of BVND and EBVND under unbalanced traffic load

Algorithm	Average Cell Delay (timeslot)	Variance of Cell Delay (timeslot*timeslot)	Average Queue Length (cell)
BVND	826	1.1e+5	31
WFBVND-$logN$	296	8.3e+4	11
WFBVND	22.6	4.1e+3	0.43

CHAPTER 6
THE CREDIT-BASED ALGORITHMS

Both of the BVND and EBVND algorithms have very high off-line computational complexity. Whenever the configuration of connections is changed, BVND and EBVND must recalculate the doubly stochastic matrix and decompose it. The complexity of the construction and decomposition of the doubly stochastic matrix makes BVND and EBVND difficult to be implemented. In this chapter, we will introduce several new credit-based algorithms which have lower off-line computational complexity with the similar QoS features to BVND.

6.1 The Algorithms

Suppose every $Q_{i,j}$ has a credit $c_{i,j}$ which is a real variable that has the initial value of 0. At the beginning of every timeslot, $c_{i,j}$ increases by $\tilde{r}_{i,j}$, where $\tilde{R} = (\tilde{r}_{i,j})$ is the doubly stochastic matrix defined in the last chapter. The scheduler selects the match according to the current credits of VOQs. Define the matching matrix at timeslot n as $\mathbf{S}(n) = (S_{i,j}(n))$, where $S_{i,j}(n)$ equals to 1 if input-output pair (i, j) is in the match, otherwise $S_{i,j}(n) = 0$. Only permutation matrices are considered as the matching matrix $\mathbf{S}$, implying that there are always exactly N pairs in every match. From all the permutation matrices, the maximum credit first (MCF) algorithm selects the one that can maximize the aggregate credit, i.e.,

$$\mathbf{S}(n) = \arg\max_{\mathbf{S}}[\sum_{i,j} S_{i,j}(n)c_{i,j}(n)],$$

where, $\sum_j S_{i,j}(n) = \sum_i S_{i,j}(n) = 1, \forall i, j$. The matching matrix $\mathbf{S}$ can be found by MWM algorithm which has a computational complexity of $O(N^3)$ [66]. The cells in VOQs are transmitted across the fabric according to their corresponding values in the matching matrix $\mathbf{S}$: if $S_{i,j}$ equals to 1 and $Q_{i,j}$ is not empty, then the HOL cell of $Q_{i,j}$ will be transmitted to output j. In the meanwhile, $c_{i,j}$ decreases by $S_{i,j}$ for all i, j. Thus, $c_{i,j}(n)$, the credit of $Q_{i,j}$ at the end of timeslot n, is

$$c_{i,j}(n) = c_{i,j}(n-1) - S_{i,j}(n) + \tilde{r}_{i,j}. \tag{6.1}$$

where, $c_{i,j}(n-1)$ is the credit of $Q_{i,j}$ at the end of timeslot $n-1$.

MCF algorithm shares the same problem with BVND algorithm: the efficiency. Since MCF selects the permutation matrix solely according to the credits of VOQs and pays no attention to the queue length, the VOQs selected by MCF may be empty while other contention-free VOQs have cells. Enhanced MCF (EMCF) algorithm is introduced to solve this problem. It fills the "holes" by scanning the VOQs with the selected permutation matrx. Similar to WFBVND, it completes the scanning in N phases. Suppose $\mathbf{P}$ is the permutation matrix selected by MCF. In the lth phase, EMCF calculates matrix $\mathbf{V}_l = (P_{(i+l)modN,j})$. Then, it checks the VOQs corresponding to the non-zero elements of $\mathbf{V}_l$ and adds the non-empty VOQs in the match if both its input and output are available.

Similar to the proof of Theorems 4 and 5, the following theorems can be readily proved.

Theorem 6 *If the exactly same traffic is fed into switch $\mathcal{M}$, the IQ switch using MCF algorithm, and $\mathcal{N}$, the IQ switch using EMCF algorithm, concurrently and no cell is dropped, then $L^{\mathcal{N}}_{i,j}(n)$, the queue length of $Q_{i,j}$ in switch $\mathcal{N}$ at timeslot n, is always less than or equal to $L^{\mathcal{M}}_{i,j}(n)$ for any i, j, and n.*

Theorem 7 *Assume all the VOQs in switch $\mathcal{M}$ and $\mathcal{N}$ are FIFOs. If the exactly same traffic is fed into switch $\mathcal{M}$ and $\mathcal{N}$ concurrently and no cell is dropped, then $D^{\mathcal{N}}_{\mathbf{c}}$, the delay time of cell $\mathbf{c}$ in switch $\mathcal{N}$, is always less than or equal to $D^{\mathcal{M}}_{\mathbf{c}}$ for any cell $\mathbf{c}$.*

Theorems 6 and 7 proved the performance of EMCF is better than that of MCF in terms of cell delay and buffer length. The complexity of MCF is $O(N^3)$ owing to MWM. The complexity of the filling process is $O(N^2)$. Thus, the complexity of EMCF is $O(N^3)$, which is at the same level as MCF.

Owing to the $O(N^3)$ complexity, MCF and EMCF algorithms are difficult to implement in high speed networks, so an iterative approximation of MCF algorithm is proposed: iterative maximum credit first (IMCF) algorithm. IMCF performs the following three steps in every iteration:

1. Request: Each unmatched input sends a request to every output.
2. Grant: If an unmatched output receives any requests, it chooses the one with the largest credit. Ties are broken randomly.
3. Accept: If an input receives any grants, it chooses the one with the largest credit. Ties are broken randomly.

IMCF stops when there is no unmatched input and output. It converges in at most N iterations. The matching matrix $\mathbf{S}$ selected by IMCF is always a permutation matrix.

Property 1 *At the end of any timeslot, the credits of an IQ switch using MCF or IMCF satisfy the following equations:*

$$\begin{cases} \sum_{j=0}^{N-1} c_{i,j} = 0, & \forall i \\ \sum_{i=0}^{N-1} c_{i,j} = 0, & \forall j \\ \sum_{i,j} c_{i,j} = 0 \end{cases} . \tag{6.2}$$

Property 2 *In any timeslot just before the matching matrix is calculated, the credits of an IQ switch using MCF or IMCF satisfy the following equations:*

$$\begin{cases} \sum_{j=0}^{N-1} c_{i,j} = 1, & \forall i \\ \sum_{i=0}^{N-1} c_{i,j} = 1, & \forall j \\ \sum_{i,j} c_{i,j} = N \end{cases} . \tag{6.3}$$

Table 6.1 Maximum and minimum credit of MCF and IMCF

N	16			32			64		
U	226			962			3970		
Algorithm	c^+	c^-	Δ_c	c^+	c^-	Δ_c	c^+	c^-	Δ_c
MCF	1.08	−1.07	2.15	1.05	−0.96	2.01	0.95	−0.98	1.93
IMCF	1.36	−3.69	5.05	1.06	−4.15	5.21	1.16	−3.35	4.51

6.2 Discussion and Simulations

The credit of $Q_{i,j}$ at timeslot n can be expressed as

$$c_{i,j}(n) = \tilde{r}_{i,j} \cdot n - C_{i,j}(n). \tag{6.4}$$

For any $n \geq m$,

$$C_{i,j}(n) - C_{i,j}(m) = \tilde{r}_{i,j}(n-m) - c_{i,j}(n) + c_{i,j}(m). \tag{6.5}$$

If we assume the lower bound and upper bound of the credit are c^- and c^+, respectively, and let $\Delta_c = c^+ - c^-$, then

$$r_{i,j}(n-m) - \Delta_c \leq C_{i,j}(n) - C_{i,j}(m) \leq r_{i,j}(n-m) + \Delta_c. \tag{6.6}$$

Eq. (6.6) implies that if $A_{i,j}$, the traffic from input i to output j, conforms to $(\sigma_{i,j}, r_{i,j})$, then the cell delay from input i to output j is bounded by $\lceil(\sigma_{i,j}+\Delta_c)/r_{i,j}\rceil$.

Comparing Eq. (6.6) with Eq. (5.5), we can say if the Δ_c of MCF and IMCF is comparable with $U = N^2 - 2N + 2$, then MCF and IMCF are as good as BVND in terms of QoS guarantees. Simulation results of c^+, c^-, and Δ_c are shown in Table 6.1. The simulations are made under various non-uniform rate reservations on input-queued switches with $N = 16$, $N = 32$, and $N = 64$. Table 6.1 shows the Δ_c of MCF and IMCF, which are much smaller than U.

Both BVND and MCF try to keep the credits of VOQs as close to 0 as possible to ensure fairness. Thus, $H(n) = \sum_{i=0,j=0}^{N-1,N-1} c_{i,j}^2(n)$ can be used as the measurement

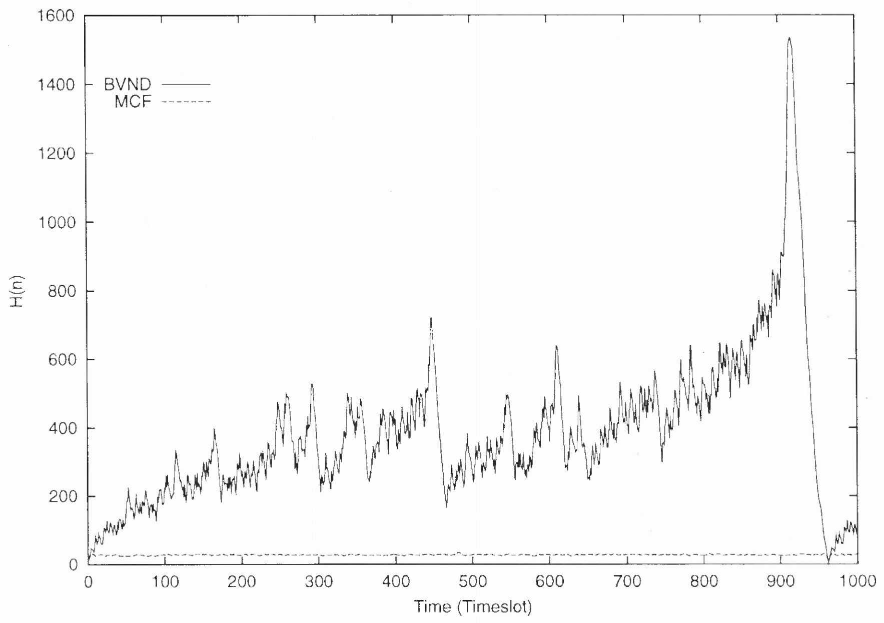


Figure 6.1 $H(n)$ of BVND and MCF under unbalanced traffic reservation.

of the fairness. For an ideal scheduling algorithm in fluid model, $H(n)$ should be equal to 0 at any timeslot, which is not possible in the packetized case. In practice, an algorithm with better fairness should have a smaller $H(n)$. From Eq. 6.1, we have

$$H(n+1) - H(n) = N - \sum \tilde{r}_{i,j}^2 + 2\sum c_{i,j}(n)\tilde{r}_{i,j} - 2\sum c'_{i,j}(n)S_{i,j},$$

where $c'_{i,j}(n) = c_{i,j}(n) + \tilde{r}_{i,j}$. Since MCF selects the matching matrix $\mathbf{S}$ which has the maximum $\sum c'_{i,j}(n)S_{i,j}$. Therefore, MCF can minimize $H(n)$ at any step. The comparison of BVND and MCF under an unbalanced traffic reservation for a 16×16 switch is shown in Fig. 6.1. In this example, the maximum $H(n)$ of BVND and MCF are 1550 and 30 respectively, implying that MCF has much better fairness than BNVD.

Figure 6.2 and 6.3 show the distribution of the percentage of cells which experience various delays over a 16×16 IQ switch using MCF, IMCF and BVND

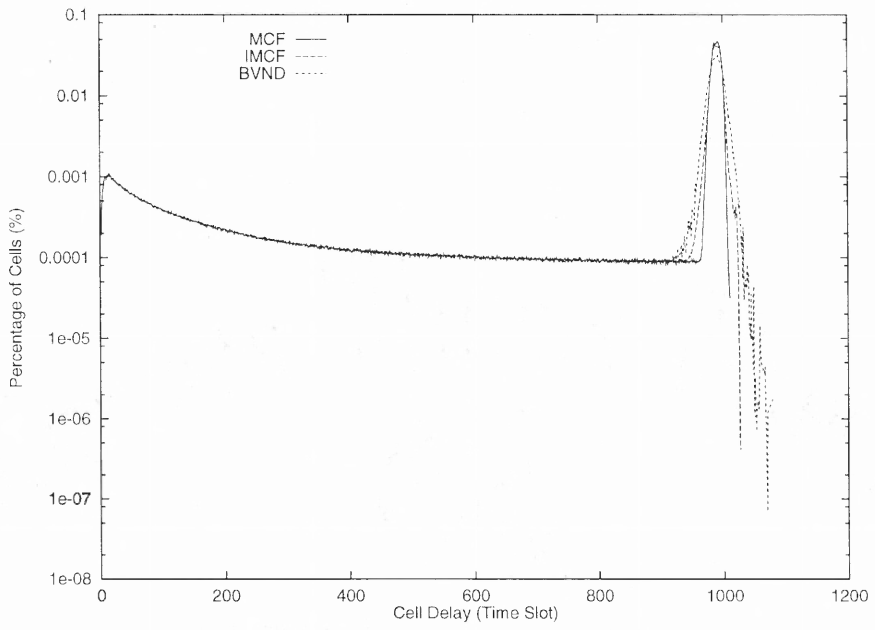


Figure 6.2 Cell delay distribution under (σ, r) traffic using MCF, IMCF, and BVND.

under non-uniform traffic with a total traffic load of 90%. Traffic $A_{i,j}$ conforms to $(\sigma_{i,j}, r_{i,j})$ for all i, j, in which $\sigma_{i,j} = 1000 r_{i,j}$. Thus, the designed delay bound is $1000 + \lceil u_{i,j}/r_{i,j} \rceil$ timeslots for BVND and $1000 + \lceil \Delta_c / r_{i,j} \rceil$ timeslots for MCF and IMCF. The simulation result shows that the cell delay bound is less than 1200 timeslots for all the algorithms as we have expected. Figure 6.2 and 6.3 demonstrates that the performance of BVND and IMCF are almost identical, and MCF has a tighter bound than the other two algorithms.

When $r_{i,j}$, the reserved rate between input i and output j, changes, MCF and IMCF need to recalculate $\tilde{\mathbf{R}}$. On the other hand, BVND not only need to recalculate $\tilde{\mathbf{R}}$, but also need to decompose it again. It has the complexity of $O(N^{4.5})$ and is termed as off-line computational complexity in [6]. Owing to the high off-line computational complexity, BVND will be hard to implement in a dynamic environment. Table 6.2 compares the computational and memory complexity of BVND, WFBVND, WFBVND-$logN$, MCF, EMCF, and IMCF. It indicates that compared to BVND

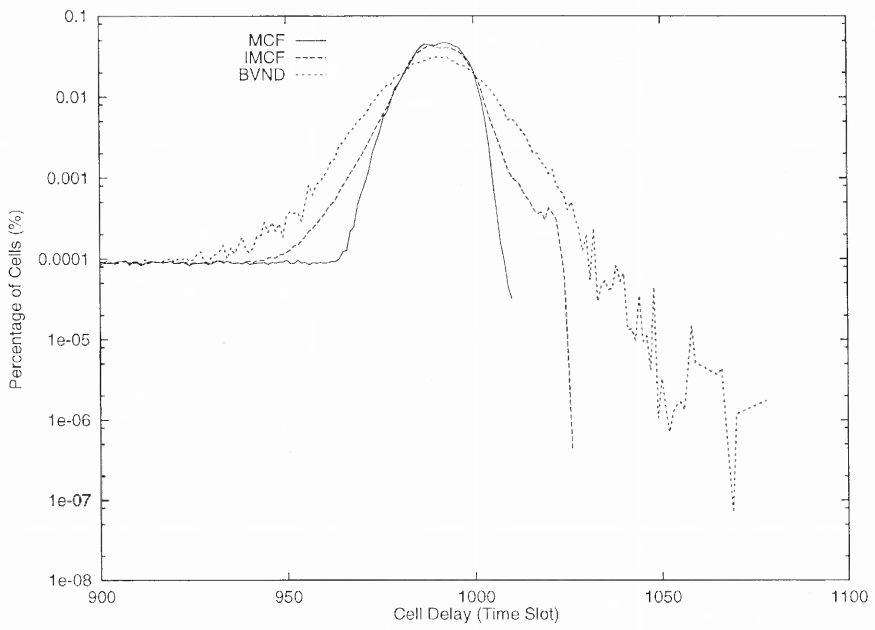


Figure 6.3 Cell delay distribution under (σ, r) traffic using MCF, IMCF, and BVND (portion).

algorithm, MCF, EMCF and IMCF have higher on-line computational complexity but have lower off-line computational complexity and on-line memory complexity.

Figure 6.4 shows the distribution of the percentage of cells which experience various delays over the switch using MCF and EMCF algorithms under unbalanced $\sigma, r)$ traffic with an average traffic load of 79%, in which $r_{i,j}$ is selected randomly, $\sigma_{i,j}$ is set to be $1000r_{i,j}$. In this example, the traffic was generated with $on - off$ model and constrained by leaky bucket. This figure demonstrates that EMCF has a smaller delay bound than MCF. Under this traffic conditions, the average cell delay of MCF is 850 timeslots. On the other hand, the average cell delay of EMCF is only 51, which is much smaller than MCF. Figure 6.5 shows the average cell delay of BVND and EMCF versus the traffic load under *i.i.d.* (σ, r) arrival. It indicates that EMCF reduces the average cell delay significantly. Figure 6.6 shows the variance of cell

Table 6.2 Computational and memory complexity of BVND-based algorithms

Algorithm	off-line computational	on-line computational	on-line memory
BVND	$O(N^{4.5})$	$O(logN)$	$O(N^3logN)$
WFBVND	$O(N^{4.5})$	$O(N^2)$	$O(N^3logN)$
WFBVND-$logN$	$O(N^{4.5})$	$O(NlogN)$	$O(N^3logN)$
MCF	$O(N^2)$	$O(N^3)$	$O(N^2)$
IMCF	$O(N^2)$	$O(N^2)$	$O(N^2)$
EMCF	$O(N^2)$	$O(N^3)$	$O(N^2)$

delay versus the traffic load under the same traffic model as in Fig. 6.5. Compared with BVND, the variance of cell delay of EMCF is also much smaller.

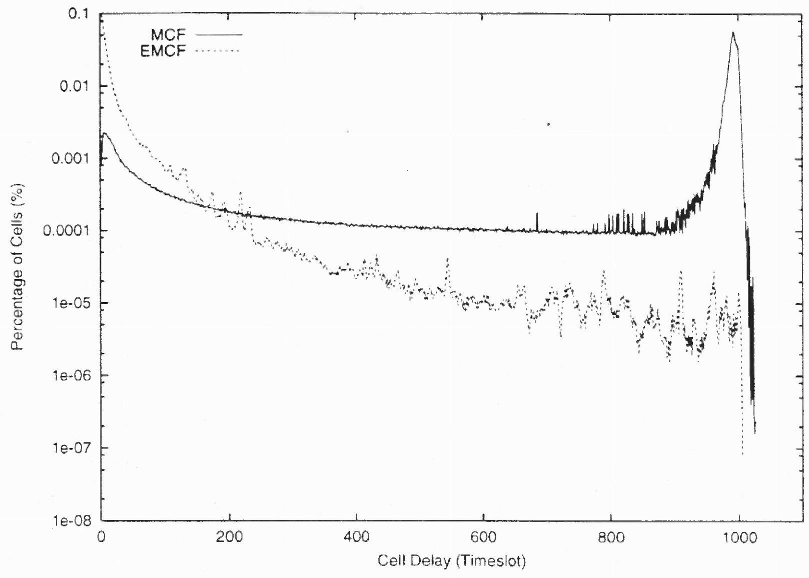


Figure 6.4 Cell delay distribution under unbalanced (σ, r) traffic using MCF and EMCF algorithms.

In summary, the three new algorithms, MCF, EMCF, and IMCF, which have similar performance as BVND in terms of QoS guarantees, but have less off-line computational and on-line memory complexity, which makes these new algorithms more realizable in practice. Simulations also show that the fairness of MCF is much better than that of BVND.

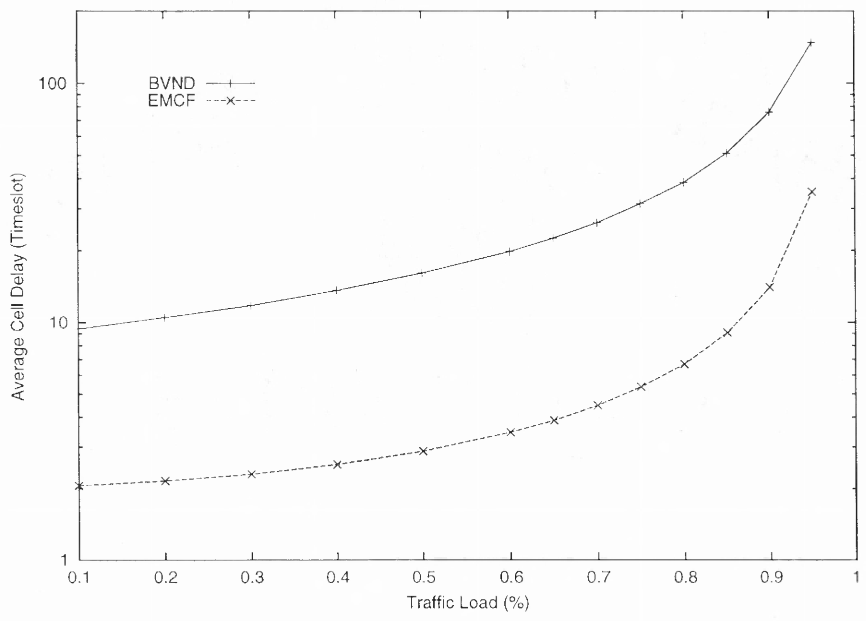


Figure 6.5 Average cell delay vs. traffic load under *i.i.d.* (σ, r) traffic using BVND and EMCF algorithms.

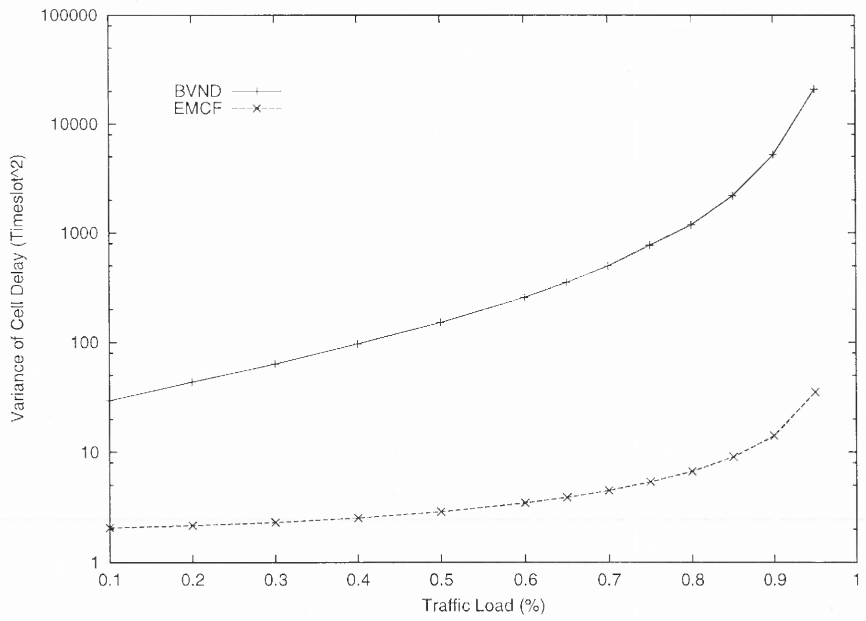


Figure 6.6 Variance of cell delay vs. traffic load under *i.i.d.* (σ, r) traffic using BVND and EMCF algorithms.

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

In this dissertation, the scheduling algorithms for high speed VOQ switches were studied. Several new VOQ scheduling algorithms were proposed and evaluated. It was shown by simulations as well as theoretical analyses that the new algorithms can achieve better QoS features than the existing algorithms.

Earliest due date first matching (EDDFM), a MWM algorithm, is proved analytically to be stable and starvation-free under all admissible independent traffic patterns. Simulations show that EDDFM has lower probability of cell overdue than other MWM algorithms such as LNQF, LQF, and OCF.

Iterative shadow departure time algorithm (ISDTA), which employs maximal matching and thus has a lower complexity than algorithms which use MWM, has been proposed to improve upon existing maximal matching algorithms in terms of QoS features. The cell delay distribution using i-LQF exhibits a heavy tail especially under the bursty traffic implying that more cells experience much longer latency than those using i-OCF and ISDTA. Simulations also show that ISDTA has a larger cumulative distribution of cell delay than i-OCF, which implies that switches using ISDTA have a lower probability of overdue than those using i-OCF. It has been shown by simulations that the performance of a CIOQ switch using ISDTA with a speedup of 1.5 is similar to that of a OQ switch in terms of cell delay and throughput. An unsolved problem is that whether a delayed PIFO-OQ switch is able to be exactly mimicked by a VOQ switch with the speedup less than 2.

The enhanced Birkhoff-von Neumann decomposition (EBVND) algorithm is based on BVND. Theoretical analysis indicates that the performance of EBVND is better than BVND in terms of throughput and cell delay, which indicates that it can also provide rate and cell delay guarantees. Wave front Birkhoff-von Neumann decomposition (WFBVND) algorithm and its simplified version WFBVND with *logN* iterations (WFBVND-*logN*), the special cases of EBVND, are also introduced and evaluated. Simulations show that WFBVND and WFBVND-*logN* have much lower average cell delay comparing to BVND.

The maximum credit first (MCF) algorithm is introduced to provide rate and cell delay guarantees. The iterative maximum credit first (IMCF) algorithm is the simplified version of MCF. Simulations show that both MCF and IMCF have similar performance as the Birkhoff-von Neumann decomposition (BVND) algorithm which can provide cell delay bound and 100% throughput but lower off-line computational and on-line memory complexity. Enhanced MCF (EMCF) algorithm is also presented in this dissertation to improve the efficiency of MCF. EMCF has a much smaller average cell delay than MCF and BVND.

Simulation results illustrate that the performance of a CIOQ switch with speedup of 1.5 is almost the same as a FIFO-OQ switch in terms of throughput and cell delay. However, whether a delayed OQ switch is able to be exactly mimicked by a CIOQ switch with the speedup less than $2 - 1/N$ needs to be investigated further. Simulations reveal that the bound of credits under MCF is tighter than that of BVND, which implies that the MCF has a smaller delay bound and better fairness than that of BVND. The question is whether we can identify the bound of MCF via analytical work. Our future effort will be focusing on answering these questions.

REFERENCES

1. T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Computer Systems,* vol. 11, no. 4, pp. 319-352, Nov. 1993.

2. The ATM Forum, "Traffic Management Specification," *AF-TM-0121.000,*Version 4.1, Mar. 1999.

3. J.C.R. Bennett and H. Zhang, "$\mathrm{WF}^2\mathrm{Q}$: worst-case fair weighted fair queueing", *Proc. INFOCOM'96,* San Francisco, CA, Mar. 1996, pp. 120-128.

4. R. Bolla, F. Davoli, and M. Marchese, "Evaluation of a cell loss rate computation method in ATM multiplexers with multiple bursty sources and different traffic classes," *Proc. GLOBECOM'96*, pp. 437-441, Nov. 1996.

5. T.X. Brown, H. Gabow, "Future information in input queueing," submitted to *IEEE Trans. Commum.*.

6. C.S. Chang, W.J. Chen, and H.Y. Huang, "On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," *Proc. IEEE IWQoS'99,* London, U.K., 1999, pp.79-86.

7. C.S. Chang, W.J. Chen, and H.Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," *Proc. INFOCOM 2000,* Tel Aviv, Israel, Mar. 2000, pp. 1614-1623.

8. C.Y. Chang, A.J. Paulraj, and T. Kailatch, "A broadband packet switch architecture with input and output queueing," *Proc. Globecom'94,* 1994, pp. 448 - 452.

9. A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbar with speedup," *Proc. 6th IEEE/IFIP IWQoS'98,* Napa, CA, May 1998.

10. J.S. Chen and T.E. Stern, "Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch," *IEEE J. Select. Areas Commum.,* vol. 9, pp. 439-449, Apr. 1991

11. S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," *Computer Systems Technical Report CSL-TR-98-758,* Mar. 1998.

12. S.T. Chuang, A. Goel, N. Nckeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE J. Select. Areas Commun.,* vol. 17, No. 6, pp. 1030-1039, June 1999.

13. T. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithm,* The MIT Press, Cambridge, Massachusetts, Mar. 1990.

14. J.G. Dai, B. Prabhakar, "The throughput of data switches with and without speedup," *Proc. INFOCOM 2000,* Tel Aviv, Israel, Mar. 2000.

15. A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm," *Internet. Res. and Exper.,* vol. 1, 1990.

16. A. Diwan, R. Guerin, and K. N. Sivarajan, "Performance analysis of speeded-up high-speed packet switches," *Proc. BC'99,* Hong Kong, Nov. 1999.

17. H. Duan, J.W. Lockwood, S.M. Kang, and J.D. Will, "A high-performance OC-12/OC-48 queue design prototype for input-buffered ATM switches," *Proc. INFOCOM'97,* 1997.

18. H. Duan, and J.W. Lockwood, S.M. Kang, "Matrix unit cell scheduler (MUCS) for input-buffered ATM switches," *IEEE Communications Letters,* vol. 2, no. 1, pp. 20-23, Jan. 1998.

19. D. Gale and L.S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly,* vol. 69, pp. 9-15, 1962.

20. S. Golestani, "A stop-and-go queueing framework for congestion management," *Proc. ACM SIGCOMM'90,* Philadelphia, PA, Sept. 1990, pp. 8-18.

21. S. Golestani, "A self-clocked fair queueing scheme for broadband applications," *Proc. INFOCOM'94,* 1994, pp. 636-646.

22. M.W. Goudreau, S.G. Kolliopoulos, and S.B. Rao, "Scheduling algorithms for input-queued switches: Randomized techniques and experimental evaluation," *Proc. INFOCOM 2000,* Tel Aviv, Israel, Mar. 2000.

23. R.A. Guerin and K.N. Sivarajan, "Delay and throughput performance of speeded-up input-queueing packet switches," *IBM Res. Rep. RC 20892,* 1997.

24. A.L. Gupta and N.D. Georganas, "Analysis of a packet switch with input and output buffers and speed constraints," *Proc. InfoCom'91,* Bal Harbour, FL, 1991, pp. 694-700.

25. J.E. Hopcroft and R.M. Karp, "An $n^{(5/2)}$ algorithm for maximum matching in bipartite graphs," *Society for Industrial and Applied Mathematics J. Comput.,* vol. 2, pp. 225-231, 1973.

26. J. Hui, *Switching and Traffic Theory for Integrated Broadband Networks,* Kluwer Academic Press, 1990.

27. I. Iliadis and W. E. Denzel, "Performance of packet switches with input and output queueing," *Proc. ICC'90,* Atlanta, GA, 1990, pp. 747-753.

28. A. Kam and K.-Y. Siu, "Linear Complexity Algorithms for QoS Support in Input-Queued Switches with no Speedup," *IEEE J. Select. Areas Commun.,* vol. 17, no. 6, June 1999.

29. M. Karol, M. Hluchyi, and S. Mogan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.,* vol. COM-35, pp. 1347-1356, Dec. 1987.

30. P. Krishna, N. Patel, A. Charny, R. Simcoe, "On the speedup required for work-conserving crossbar switches," *Proc. 6th IEEE/IFIP IWQoS'98,* Napa, CA, May 1998.

31. P. Krishna, N.S. Patel, A. Charny, and R.J. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE J. Select. Areas Commun.,* vol. 17, no. 6, pp. 1057-1066, June 1999.

32. B. Kwon, B. Kim, and H. Yoon, "Self-firing cell scheduler for input queueing ATM switches," *Electronics Letters,* vol. 32, no. 17, Aug. 1996.

33. R.O. Lamaire and D.N. Serpanos, "Two dimensional Round Robin Schedulers for Packet switches with multiple input queues," *IEEE/ACM Trans. Networking,* pp. 471-482, Oct. 94.

34. S. Li, and N. Ansari, "Scheduling input-queued switches with QoS features," ICCCN'98, Lafayette, Louisiana, Oct. 1998, pp. 107-112.

35. S. Li and N. Ansari, "Input queued switching with QoS guarantees," *Proc. IEEE INFOCOM'99,* New York, NY, Mar. 1999, pp. 1152-1159.

36. S.Q. Li, "Performance of a nonblocking space-division packet switch with correlated input traffic," *Proc. GLOBECOM'89,* 1989, pp. 1754-1763.

37. S.Y. Li, "Theory of periodic contention and its application to packet switching," *Proc. INFOCOM'88,* Mar. 1988, pp. 320-325.

38. S.Y. Liew, S.W. Cheng, and T.T. Lee, "An enhanced iterative scheduling algorithm for ATM input-buffered switch," *Proc. ATM Workshop,* 1999, pp. 103-108.

39. M.A. Marsam, A. Bianco, and E. Leonardi, "RPA: A simple, efficient, and flexible policy for input buffered ATM switches," *IEEE Communications Letters,* vol. 1, no. 3, pp. 83-86, May 1997.

40. M.A. Marsan, A. Bianco, E. Leonardi, and L. Milia, "Quasi-optimal algorithms for input buffered ATM switches," *Proc. ISCC'98*, 1998, pp. 336-342.

41. N. McKeown, P. Varaiya, and J. Walrand, "Scheduling Cells in an Input-Queued Switch," *IEE Electronics Letters,* vol. 29, no. 25, pp. 2174-2175, Dec. 1993.

42. N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, Univ. of California, Berkeley, 1995.

43. N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Proc. INFOCOM'96,* San Francisco, CA, Mar. 1996, pp. 296-302.

44. N. McKeown, B. Prabhakar, and M. Zhu, "Matching output queueing with combined input and output queueing," *Proc. 35th Ann. Allerton Conf. Commun., Control. and Computing,* Monticello, Illinois, Oct. 1997.

45. N. McKeown and T.E. Anderson "A Quantitative Comparison of Scheduling Algorithms for Input-Queued Switches," *Computer Networks and BISDN Systems,* vol. 30, no. 24, pp. 2309-2326, Dec. 1998.

46. N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking,* vol. 7, no. 2, pp. 188-201, Apr. 1999.

47. N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.,* vol. 47, no. 8, pp. 1260-1267, Aug. 1999.

48. A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," *Proc. ICCCN'96,* Oct. 1996, pp. 226-231.

49. A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in Input-Queued Switches," *Proc. INFOCOM'98,* San Francisco, CA, Mar. 1998, pp. 792-799.

50. S. Motoyama, D.W. Petr, and V.S. Frost, "Input-queued switch based on a scheduling algorithm," *Electronics Letters,* vol. 31, no. 14, pp. 1127-1128, July 1995.

51. S. Motoyama, L.M. Ono, and M.C. Mavigno, "Performance analysis of iterative scheduling algorithms for ATM input-queued switches," *Proc. ITS'98,* 1998, pp. 195-200.

52. S. Motoyama, L.M. Ono, and M.C. Mavigno, "An iterative cell scheduling algorithm for ATM input-queued switch with service class priority," *IEEE Communications Letters,* vol. 03, no. 11, pp. 323-325, Nov. 1999.

53. N. Ge, J.K. Muppala, and M. Hamdi, "Analysis of non-blocking ATM switches with multiple input queues," *GLOBECOM'97,* 1997, pp. 531-535.

54. Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch," *Proc. ICC'89,* Boston, MA, 1989, pp. 410-414.

55. A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE Trans. Networking,* vol. 1, no. 3, pp. 344-357, June 1993.

56. A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case", *IEEEACM Trans. Networking,* vol. 2, no. 2, pp. 137-150, Apr. 1994.

57. B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," *Computer Systems Technical Report CSL-TR-97-738,* Nov. 1997.

58. T.L. Rodeheffer, J.B. Saxe, "An efficient matching algorithm for a high-throughput, low-latency data switch," *Compaq Comput. Corp., Syst. Res. Ctr., Res. Rep.,* 1998.

59. R. Schoenen, G. Post, G. Sander, "Prioritized arbitration for input-queued switches with 100% throughput," *Proc. ATM Workshop,* 1999, pp. 253-258.

60. D.N. Serpanos and P.I. Antoniadis, "FIRM: A class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues," *Proc. INFOCOM'96,* San Francisco, CA, Mar. 1996.

61. D. Stiliadis and A. Varma, "Providing Bandwidth Guarantees in an Input Buffered Crossbar Switch," Proc. *Proc. INFOCOM'95,* Boston, MA, Apr. 1995.

62. I. Stoica and H. Zhang "Exact emulation of an output queueing switch by a combined input output queueing switch," *Proc. IWQoS'98,* 1998.

63. V. Tabanaee, L. Georgiadis, and L. Tassiulas, "QoS Provisioning and track fluid policies in input queueing switches," *Proc. INFOCOM 2000,* Tel Aviv, Israel, Mar. 2000.

64. Y. Tamir and G.L. Frazier, "High-performance multi-queue buffers for VLSI communication switches," *Proc. 15th Annu. Int. Symp. Comput. Architecture,* Honolulu, HI, May 1988, pp. 343-354.

65. Y. Tamir and H.C. Chi "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 1, pp. 13-27, Jan. 1993.

66. R.E. Tarjan, *Data structures and network algorithms,* Society for Industrial and Applied Mathematics, Pennsylvania, Nov. 1983.

67. L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio

networks," *IEEE Trans. Automatic Control,* vol. 37, no. 12, pp. 1936-1948, Dec. 1992.

68. Y.-S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: A simple modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.,* vol. SAC-5, pp. 1274-1283, Oct. 1987.

69. H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE,* vol. 83, no. 10, pp. 1374-1396, Oct. 1995.