ABSTRACT

ON SCHEDULING INPUT QUEUED CELL SWITCHES

by
Shizhao Li

Output-queued switching, though is able to offer high throughput, guaranteed delay and fairness, lacks scalability owing to the speed up problem. Input-queued switching, on the other hand, is scalable, and is thus becoming an attractive alternative. This dissertation presents three approaches toward resolving the major problem encountered in input-queued switching that has prohibited the provision of quality of service guarantees.

First, we proposed a maximum size matching based algorithm, referred to as min-max fair input queueing (MFIQ), which minimizes the additional delay caused by back pressure, and at the same time provides fair service among competing sessions. Like any maximum size matching algorithm, MFIQ performs well for uniform traffic, in which the destinations of the incoming cells are uniformly distributed over all the outputs, but is not stable for non-uniform traffic. Subsequently, we proposed two maximum weight matching based algorithms, longest normalized queue first (LNQF) and earliest due date first matching (EDDFM), which are stable for both uniform and non-uniform traffic. LNQF provides fairer service than longest queue first (LQF) and better traffic shaping than oldest cell first (OCF), and EDDFM has lower probability of delay overdue than LQF, LNQF, and OCF. Our third approach, referred to as store-sort-and-forward (SSF), is a frame based scheduling algorithm. SSF is proved to be able to achieve strict sense 100% throughput, and provide bounded delay and delay jitter for input-queued switches if the traffic conforms to the (r, T) model.

ON SCHEDULING INPUT QUEUED CELL SWITCHES

by
Shizhao 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

Department of Electrical and Computer Engineering

May 1999

APPROVAL PAGE

ON SCHEDULING INPUT QUEUED CELL SWITCHES

Shizhao Li

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

Dr. John Carpinelli, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Xiaoqiang Chen, Committee Member — Date
Member of Technical Staff, Bell Labs, Lucent Technologies, Holmdel, NJ

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

Dr. Bulent Yener, Committee Member — Date
Assistant Professor of Computer and Information Science, NJIT

BIOGRAPHICAL SKETCH

Author: Shizhao Li

Degree: Doctor of Philosophy

Date: May 1999

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1999

- Master of Science in Electrical Engineering,
  Beijing University of Posts and Telecommunications, Beijing, P. R. China, 1994

- Bachelor of Science in Electrical Engineering,
  Shandong Univeristy, Jinan, Shandong, P. R. China, 1991

Major: Electrical Engineering

Presentations and Publications:

- S. Li, J. G. Chen, and N. Ansari, "Fair queueing for input-buffered ATM switches," ICATM'98, France, pp. 252-259, Jun., 1998.

- R. Venkateswaran, S. Li, X. Chen, C.S. Raghavendra, and N. Ansari, "Enhanced VC merging mechanisms for multipoint to multipoint communications," ICCCN'98, Lafayette, Louisiana, pp. 4-11, Oct. 1998.

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

- S. Li, and N. Ansari, "Provisioning QoS features for input-queued switches," IEE Electronics Letters, vol. 34, no. 19, pp. 1826-1827, Sept. 17.

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

- S. Li, J. Li, and N. Ansari, "Earliest Due Date First Matching for Input-Quered Cell Switches," CISS'99, Baltimore, MD, Mar. 1999.

This work is dedicated to
my family

ACKNOWLEDGMENT

I would like to thank my advisor Dr. Nirwan Ansari for his continuous guidance, support and encouragement throughout my graduate study at NJIT. I greatly appreciate the knowledge and insightful comments he has been providing to me. I feel fortunate that I had a chance to conduct research under such an excellent advisor. Dr. Xiaoqiang Chen gave me so much help when I worked at Lucent Technologies as a summer intern, and he continues to give me advice. I am honored to express my gratitude to him. I would also like to thank Dr. John Carpinelli, Dr. Sirin Tekinay, and Dr. Bulent Yener for serving on my dissertation committee.

I would also like to thank Dr. Jianguo Chen for all the helpful discussions. Dr. R. Venkateswaran helped me understand many problems when I worked with him at Lucent Technologies. I thank Dr. Xueming Lin and Dr. Huaping Liu for helping me familiarize with the working environment when I first came to the Center for Communications and Signal Processing.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1
INTRODUCTION

Telecommunication networks have been evolving from pure circuit switching based telephone networks to packet switching based broadband integrated service networks providing services for transport of voice, audio, images, real-time video, graphics, data, and other multimedia applications. Different applications have strikingly different requirements of the quality of service (QoS), and thus the design of traffic scheduling for switches to satisfy these various requirements is very crucial and challenging.

1.1 Buffering Schemes and its Effects on the Design of Traffic Scheduling

The objective of traffic scheduling is to satisfy the requirements of guaranteed performance such as delay, delay jitter for real-time traffics, and a fair distribution of the network resources for best effort applications. Scheduling algorithms fulfill this task by selecting a cell for transmission in the next transmission period for each output link of the switch among the cells destining for the same output link. Switch fabric architectures and buffering mechanisms affect the scheduling algorithm design. An ATM switch typically consists of three parts: input ports, output ports and a switch fabric. Input ports buffer cells coming from input links while output ports buffer cells going out to output links. The fabric routes cells from arbitrary input links to arbitrary output links. Many architectures for switch fabrics have been proposed in the literature. Shared memory, bus, crossbar, and multistage networks [1] are among the commonly used architectures.

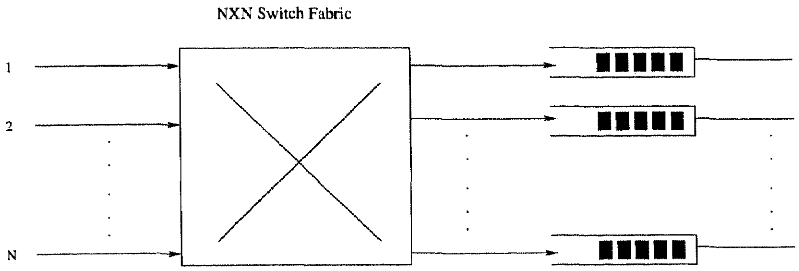


Figure 1.1 Output buffering architecture

In general, switch architectures can be categorized into three main types based on the adopted buffering mechanisms: the input-queued switch in which buffers are placed at the input side, the output-queued switch in which buffers are placed at the output side, and the combined input output queued switch in which buffers are placed at both the input and output sides.

Most of the early studies [2, 3, 4, 5, 6, 7, 8] focused on scheduling output queued switches (OQ) owing to its conceptual simplicity. By assuming that cells are readily available to be transmitted to the output links upon entering a switch, as shown in Figure 1.1, many proposed algorithms are able to provide QoS guarantees (see [9] for an overview). However, the output queueing architecture suffers from the scalability problem. Since more than one cell can arrive at the switch in a given time slot heading for the same output, the fabric and output buffers should have the capability to accommodate all of the cells to avoid cell loss and to meet certain delay bounds. In the worst case, an $N \times N$ switch has to run N times faster than a single link when all N inputs receive cells directed to the same output in a time slot. The buffers, switch fabric, and control system have to be sped up proportionally to the number of input or output links, and thus severely limiting the switch capacity.

The input queueing (IQ) architecture, on the other hand, has good scalability. Since buffers are placed at the input of the switch, as shown in Figure 1.2, the fabric

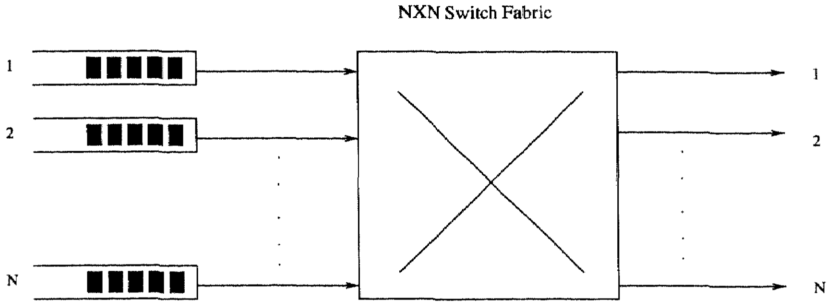


Figure 1.2 Input buffering architecture

and buffers can run at the same speed as a single link without causing cell loss. Owing to its scalability, input queueing is receiving attention in both the research and commercial communities [10, 11, 12, 13, 14, 15]. Low throughput and no QoS guarantees are two major problems with the input queueing architecture. Extensive studies [12, 13, 14, 15, 16, 17] on improving the throughput of an input-queued switch have been conducted in the literature, and studies on providing QoS guarantees are still undergoing [18, 19].

There has been a trade off between QoS guarantees and scalability: the input queueing architecture is scalable but cannot provide guaranteed QoS, while the output queueing architecture can provide guaranteed QoS but is not scalable. Lately, there is a trend to adopt combined input output queueing (CIOQ), in which buffers are placed at both the input and output sides of a switch [20, 21]. It has been proven in [20] that a speedup of 2 is sufficient for a CIOQ switch to behave identically to an output-queued switch which employs work-conserving and monotonic scheduling discipline.

1.2 Design Criteria for a Traffic Scheduler

Below is a high-level description of design criteria that switch designers must consider for the design and implementation of an appropriate traffic scheduler.

- Isolation among flows: A connection sending data at or below its negotiated rate should not be affected by misbehaving connections which send data at a rate higher than their negotiated rates.

- End-to-end delay: Real time applications have stringent requirements on the end-to-end delay. Cells arrived too late may have no use to the applications. Thus, scheduling algorithms should be able to provide guaranteed end-to-end delay bound for individual sessions.

- End-to-end delay jitter: The end-to-end delay jitter of a session is defined as the maximum difference of the delay experienced by any two cells belonging to the session. Continuous media playback applications usually have stringent requirements on the delay jitter. Ideally, a network should not introduce any delay jitter for these applications (i.e., constant delay or zero delay jitter).

- Throughput: Throughput is defined as the highest load under which a switch can forward without dropping any cell.

- Fairness: If the bandwidth usage of a connection is below its negotiated rate, the excess bandwidth should be distributed fairly among all connections. The commonly used fairness is defined as follows [4]: the fairness of a scheduling algorithm is the maximum difference of the normalized service received by any two sessions over the interval in which both sessions are continuously backlogged, i.e.,

$$F_S = \max_{\forall i,j,\ j \neq i} \left| \frac{W_i(t_1,t_2)}{r_i} - \frac{W_j(t_1,t_2)}{r_j} \right| . \qquad (1.1)$$

where $W_i(t_1, t_2)$ is the service session i received during time interval $[t_1, t_2]$, and r_i is the rate of session i. The smaller the amount, the fairer the algorithm is.

- Scalability: Since there could be thousands of connections sharing the same link, the algorithm must be scalable.
- Implementation complexity: The high switching speed imposes the scheduling algorithms to make decisions in a very short time. For example, at the rate of 622 Mbps, the switch has to make decision within less than $0.7\mu s$. Thus, scheduling algorithms must have a simple implementation.

1.3 Traffic Scheduling in Input-Queued Switches

One of the major problems with input-queued switches is the head-of-line (HOL) blocking, which limits the throughput of such a switch to only 58.6% under uniform Bernoulli traffic when a single FIFO queue is maintained in each input buffer [22]. The throughput can be improved by making more than one cell in the buffer accessible to the scheduler. A windowed buffer with a size of W, in which any of the first W cells in the buffer are accessible, can improve the throughput noticeably, even with a small W. Virtual output queueing, in which an individual FIFO queue corresponding to each output is maintained in each input buffer as shown in Figure 1.3, is another method to improve throughput. All the cells at the head of the FIFO queues are accessible to the scheduler, and thus HOL blocking is avoided. However, contentions still occur when multiple cells try to get through the fabric. Output contentions occur when more than one cell is directed to the same output link, and similarly, input contentions occur when more than one virtual output queues (VOQs) in the same input buffer are non-empty. The contentions limit the throughput of an

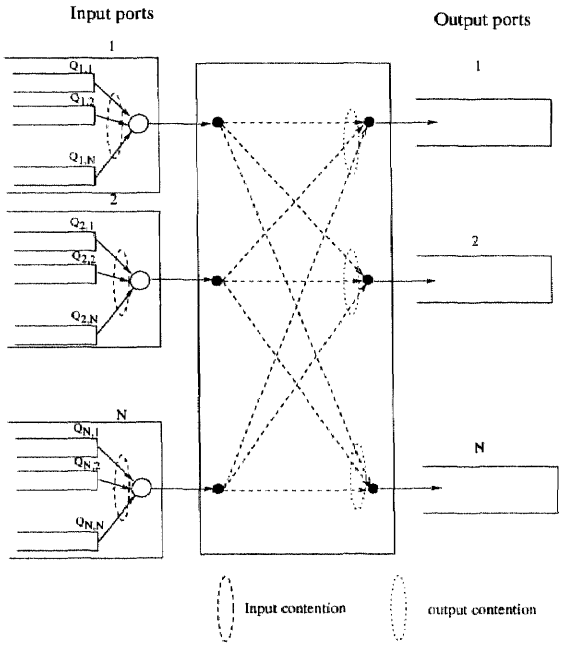


Figure 1.3 Virtual output queueing and input/output contentions

input-queued switch, and thus constrain the switch from providing QoS guarantees to the applications.

Note that when more than one cell can be accessed by the scheduler in one input buffer, selecting different cells for transmission could lead to different throughput, owing to the inter-dependence of the inputs.

Maximizing the throughput of a switch can be mapped to the maximum matching problem in a bipartite graph. Algorithms proposed in the literature can be classified into two categories: maximum size and maximum weight matching based algorithms.

1.3.1 Maximum Size Matching Based Algorithms

Maximum size matching based algorithms maximize the number of connections between the input and output ports with the constraint of unique pairing. Representative scheduling algorithms include PIM (Parallel Iterative Matching) [12],

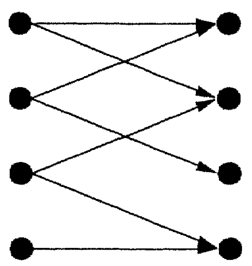


Figure 1.4 An example of a bipartite graph

*i*SLIP (Iterative Round Robin with slip) [14], and MFIQ (Min-max Fair Input Queueing) [17]. Figure 1.4 shows a bipartite graph, in which each input port is represented by a vertex in one group, and each output port is represented by a vertex in the other group. Each accessible cell is represented by an edge from an input port vertex in which it is stored to an output port vertex to which it is directed. Two possible matches of the configuration in Figure 1.4 are shown in Figure 1.5, in which one achieves the maximum throughput while the other does not.

1.3.1.1 Representative Schedulers: Finding the maximum match may take a long time [23]. Thus, most proposed mechanisms for matching resort to a maximal match, which has less complexity but in the worst case contains only 50% of the possible pairings of the maximum match [12].

Parallel Iterative Matching (PIM) is a three-phase algorithm in finding a maximal match, consisting of requests, grants, and acknowledgments between the input and output ports. The complexity of PIM is proven to converge on average in $O(logN)$ [12]. Many crossbar matching algorithms proposed in the literature [12, 24] are based on PIM.

McKeown and Anderson [14] later proposed IRRM for an ATM switch, which was found to have the same performance as PIM, but with a much lower hardware

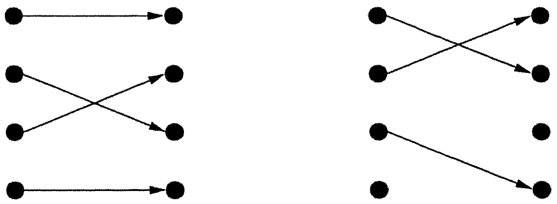


Figure 1.5 Two possible matches of a bipartite graph

complexity. Similar to PIM, the following three steps are iterated for an M input, N output switch as described in [14]:

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

2. If an unmatched output receives any requests, it chooses the one that appears next in a round-robin scheduler starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer g_i to the highest priority element of the round-robin scheduler is incremented (modulo M) to one location beyond the granted input.

3. If an input receives a grant, it accepts the one that appears next in a round-robin scheduler starting from the highest priority element. The pointer a_i to the highest priority element of the round-robin scheduler is incremented (modulo N) to one location beyond the accepted output.

An example of one iteration of the three phases is illustrated in Figure 1.6. In the example, input port 1 has one or more cells for output ports 1 and 2, input port 2 has one or more cells for output ports 1 and 3, and so on. The grant schedules are shown for output ports 1 and 2. The accept schedules are shown for input ports 1 and 2. At the end of the first iteration, g_1, g_2 and a_1 are incremented to 2, and a_2

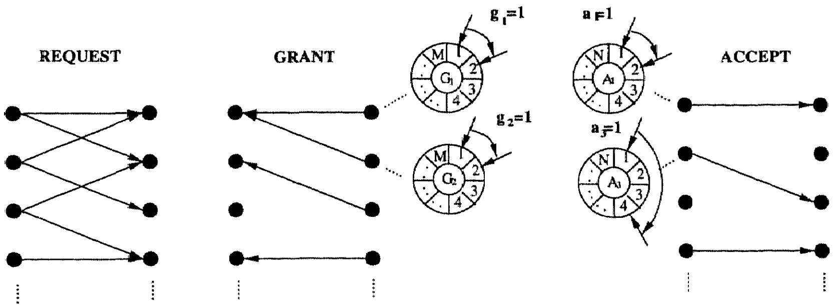


Figure 1.6 One iteration of the Iterative Round Robin Scheduling algorithm

is incremented to 4. The basic IRRM algorithm will not perform well for a single iteration because it does not allow the schedulers to be misaligned. IRRM with slip (iSLIP) was proposed later to achieve 100% utilization in one iteration. The iSLIP algorithm is identical to IRRM except that the pointer g_i in phase 2 is incremented if and only if the grant was accepted.

A 320 Gbps crossbar system is currently in development [25] adopting iSLIP and algorithms of similar nature.

1.3.2 Maximum Weight Matching Based Algorithms

Maximum size matching based algorithms work well for uniformly distributed traffic. However, it was pointed out that a switch using maximum size matching based algorithms is not stable for non-uniform traffic [13, 15], i.e., the expected queue length in the switch could increase without bound. Maximum weight matching based algorithms were proposed later to achieve 100% throughput for both uniform and non-uniform traffic.

Consider the bipartite graph shown in Figure 1.7(a). Associated with each edge is a weight, which is defined differently for different algorithms. For example, setting weight as the queue length of the VOQ leads to LQF (Longest Queue First) [13], and setting weight as the delay time of the head cell in the VOQ leads to OCF

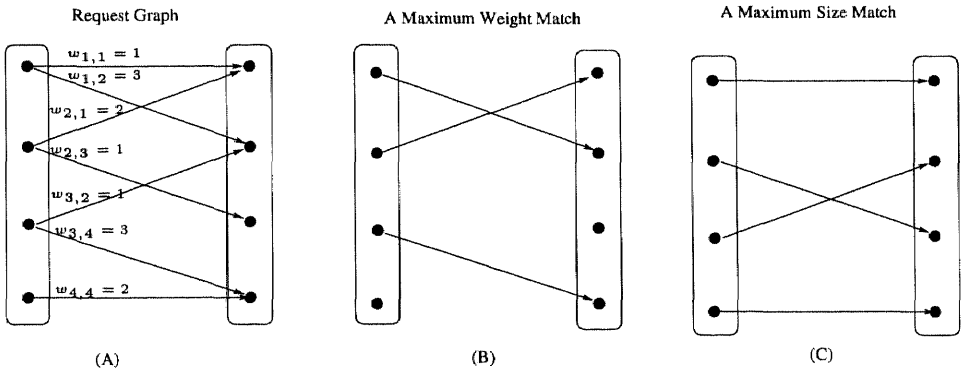


Figure 1.7 A bipartite graph matching example:(a) the request graph, (b) a maximum weight match, and (c) a maximum size match.

(Oldest Cell First) [15]. A maximum weight matching algorithm computes a match which can maximize the aggregate weight. The computational complexity of LQF and OCF is $O(N^3 logN)$. LPF [16] was proposed later to reduce the complexity to $O(N^{2.5})$, in which the weight of a VOQ $Q_{i,j}$ is set as follows:

$$w_{i,j}(n) = \begin{cases} R_i(n) + C_j(n) & L_{i,j}(n) > 0 \\ 0 & otherwise \end{cases}$$

where $L_{i,j}(n)$ is the queue length of $Q_{i,j}$ at time slot n, $R_i(n) = \sum_j^N L_{i,j}(n)$ is the total number of cells that are currently stored in input i, and $C_j(n) = \sum_i^N L_{i,j}(n)$ is the total number of cells stored at the input side destining for output j .

Note that maximum size match in which the maximum number of connections between inputs and outputs is obtained is a special case of maximum weight matching in which the weights of the non-empty VOQs are set to 1 and the weights of empty VOQs are set to 0. A maximum weight match and a maximum size match based on the same request graph are shown in Figure 1.7(b) and (c), respectively.

1.4 Traffic Scheduling in Output-Queued Switches

Since cells arrive in an output-queued switch are immediately available for transmission, the scheduler only needs to resolve the contention among the cells sharing the same output link. In general, schedulers can be classified into two types: work-conserving and non-work-conserving. A work-conserving scheduler is never idle when there are cells buffered in the system, while a non-work-conserving scheduler could be idle even if there are cells waiting in the buffer. Generalized Processor Sharing (GPS) [2], Weighted Fair Queueing (WFQ) [2], Self-Clocked Fair Queueing (SCFQ) [4], Virtual Clock [5], and Weighted Round Robin (WRR) [26] are examples of work-conserving schedulers, and Hierarchical-Round-Robin (HRR) [27], Stop-and-Go queueing [28] and Jitter-Earliest-Due-Date [29] are non-work-conserving schedulers.

1.4.1 Representative Work-conserving Schedulers

In this section, we briefly describe several representative work-conserving scheduling algorithms. We first present an idealized service discipline GPS and its packet version WFQ, also called Packetized Generalized Processor Sharing (PGPS), followed by SCFQ and Virtual Clock.

1.4.1.1 Generalized Processor Sharing: Generalized Processor Sharing (GPS) is an idealized scheduling discipline, which is defined based on a fluid-model. Associated with each session is a real number r_i, which represents its service share of the server. It is assumed that in GPS all the flows are serviced simultaneously with rates proportional to r_i, implying that the data unit is infinitely divisible. Let $B(t_1, t_2)$ be the set of sessions that are backlogged in the time interval $(t_1, t_2]$ and R be the total rate of the server. The service $W_i(t_1, t_2)$ that a backlogged session i can receive is proportional to its rate r_i. If session i is continuously backlogged during the interval $(t_1, t_2]$, the service it receives is

$$W_i(t_1, t_2) \geq \frac{r_i}{\sum_{j \in B} r_j} R(t_2 - t_1). \tag{1.2}$$

The minimum service rate offered to session i during that time interval is

$$\frac{r_i}{\sum_{j=1}^{M} r_j} R(t_2 - t_1), \tag{1.3}$$

where M is the maximum number of sessions which are backlogged in the server during that time interval. The GPS server serves each backlogged session simultaneously; each session is granted a rate equal to the minimum service rate of the session plus a fair share of the excess bandwidth which is available from sessions which are temporarily not backlogged. The normalized service time of a session, i, is defined as the amount of service $W_i(t_1, t_2)$ the session receives during that interval divided by its rate r_i. GPS, with the assumptions that data unit can be infinitely divisible (infinitesimally small) and all sessions can be served simultaneously, is a perfect scheduler with ideal fairness ($F = 0$), and low end-to-end delays.

Many Packet Fair Queueing (PFQ) algorithms [2, 3, 4] were proposed to emulate the GPS model for scheduling output-queued switches. Each PFQ algorithm maintains a system potential $v(t)$, which is initialized to be zero whenever the server becomes idle, and is updated accordingly. In addition, associated with the kth packet of a session, i, that arrives at time a_i^k, are a virtual starting time S_i^k and a virtual finishing time F_i^k. The virtual times are updated according to the following rule [2]:

$$\left\{ \begin{array}{lcl} S_i^k & = & max\{F_i^{k-1}, v(a_i^k)\} \\ F_i^k & = & S_i^k + \frac{l_i^k}{r_i} \end{array} \right. \tag{1.4}$$

where l_i^k represents the length of the kth packet of session i. PFQ algorithms approximate GPS performance by selecting cells with the minimum virtual finishing or starting time for transmission. Different policies for updating the system potential lead to different PFQ algorithms. For example, choosing real time as the system potential leads to Virtual Clock [5], and choosing the virtual finishing time of the current session in progress as the system potential leads to Self-Clocked Fair Queueing [4].

1.4.1.2 Weighted Fair Queueing: GPS cannot be implemented since servers transmit packets in their entirety, i.e., packets are not infinitesimally small, and a server cannot serve more than one packet at any time. A packet-by-packet transmission scheme called Weighted Fair Queueing (WFQ) or Packet Generalized Processor Sharing (PGPS) was proposed [2] as an approximation to GPS. The system potential used in WFQ is updated according to

$$v(t_2) \quad = \quad v(t_1) + \frac{t_2 - t_1}{\sum_{i \in B} r_i}, \tag{1.5}$$

where B represents the set of backlogged sessions during time interval $(t_1, t_2]$, during which the set of backlogged sessions are fixed. Thus, the system potential $v(t)$ is a piecewise linear functions of real time whose slopes depend on the total rate of the currently backlogged sessions.

WFQ algorithm can be summarized as follows:

1. The system potential is updated when backlogged sessions change.

2. At the arrival of a new packet, the system potential $v(t)$ is computed first. Suppose that the new arrived packet is the kth packet in session i, its virtual finishing time is updated as follows:

$$F_i^k \quad \leftarrow \quad max\{F_i^{k-1}, v(a_i^k)\} + \frac{l_i^k}{r_i}. \tag{1.6}$$

3. The scheduler selects the packet with the smallest virtual finishing time for transmission.

Let V be the maximum number of sessions a server can serve simultaneously. Maintaining a sorted list of virtual finishing time requires a computation of $O(logV)$. However, at most V events could occur during the transmission of one packet [30], and therefore, the complexity of computing virtual finishing time in WFQ algorithm is $O(V)$, making the algorithm prohibitive for implementation.

1.4.1.3 Virtual Clock: Virtual Clock (VC), proposed by Zhang [5], tries to emulate a static TDM system. In VC, the system potential is selected to be the real time and the virtual finishing time is updated according to the following rule:

$$F_i^k \leftarrow max\{F_i^{k-1}, a_i^k\} + \frac{l_i^k}{r_i}, \tag{1.7}$$

where a_i^k is the real time when the kth packet of session i arrives at the switch. The cell with the minimum virtual finishing time is selected for transmission.

The complexity of computing the virtual finishing time is reduced to $O(1)$. However, maintaining a sorted list of the virtual finishing time has a complexity of $O(logV)$, and thus, the complexity of Virtual Clock is $O(logV)$. It has been proven that VC has the same delay bound as WFQ and poor fairness, as shown in Table 1.1.

1.4.1.4 Self-Clocked Fair Queueing: Self-Clocked Fair Queueing (SCFQ) was proposed and analyzed in [4]. In SCFQ, the packet with the minimum virtual finishing time is scheduled for transmission. SCFQ chooses the virtual finishing time of the current being served packet as the system potential. Let v_{cur} be the virtual finishing time of the packet in service. The virtual finishing time of the new packet

Table 1.1 Comparison of representative work-conserving schedulers for output-queued switches. L_i is the maximum packet size of session i, L_{max} is the maximum packet size among all the sessions, R is the transmission rate of the switch, and σ_i is the bucket depth of session i.

servers	delay	fairness	complexity
WFQ	$\frac{\sigma_i+L_{max}}{r_i}+\frac{L_{max}}{R}$	$max(C_j+\frac{L_{max}}{r_i}+\frac{L_j}{r_j}, C_i+\frac{L_{max}}{r_j}+\frac{L_i}{r_i})$ $C_i = min((V-1)\frac{L_{max}}{r_i}, max(\frac{L_n}{r_n}))$	$O(V)$
VC	$\frac{\sigma_i+L_{max}}{r_i}+\frac{L_{max}}{R}$	∞	$O(logV)$
SCFQ	$\frac{\sigma_i+L_{max}}{r_i}+\frac{VL_{max}}{R}$	$\frac{L_i}{r_i}+\frac{L_j}{r_j}$	$O(logV)$

is updated as follows:

$$F_i^k \leftarrow max\{F_i^{k-1}, v_{cur}\} + \frac{l_i^k}{r_i}. \tag{1.8}$$

By simplifying the calculation of the system potential, the complexity of SCFQ is reduced to $O(logV)$. The trade off is that the end-to-end delay bounds grow linearly with the number of sessions that share the outgoing link, as shown in Table 1.1. Thus, the end-to-end delay bounds cannot be guaranteed by controlling sessions' negotiated rates.

Table 1.1 summarizes the performance of representative work-conserving output schedulers. All the three algorithms adopt leaky buckets to regulate bursty traffic, i.e., $A_i(t_1, t_2) \leq \sigma_i + \rho_i(t_2 - t_1)$, where $A_i(t_1, t_2)$ is the amount of traffic that enters the switch during any time interval $(t_1, t_2]$, ρ_i is the average sustainable rate, and σ_i is the bucket depth of session i. The fairness shown in the table is based on the assumption [31] that only two sessions i and j are compared, each of which has infinite supply of packets. As shown in Table 1.1, WFQ has the best performance but the highest complexity. Virtual Clock can provide the same delay bound as

WFQ but with poor fairness. On the contrary, SCFQ has good fairness but the delay is not bounded.

1.4.2 Representative Non-work-conserving Schedulers:

With a non-work-conserving scheduler, the server could be idle even when there are cells waiting for service. This results in a higher average packet delay and lower server throughput. However, the delay bound is a more important performance index for the guaranteed performance service [9]. Moreover, non-work-conserving schemes maintain traffic smoothness inside the network, and thus simplify the analysis of end-to-end delay in a networking environment. Therefore, non-working conserving schedulers have also been studied extensively. In this section, we briefly describe two representative non-work-conserving schedulers: Stop-and-Go and Hierarchical Round Robin.

1.4.2.1 Stop-and-Go Queueing: Stop-and-Go queueing was first introduced by Golestine [7]. A framing strategy is adopted in the algorithm to segment the time axis into fixed length periods called frames. Arriving and departing frames are defined, and a constant delay θ, where $0 \leq \theta < T$, between an arriving frame and its corresponding departing frame is introduced. Cells arrived during one arriving frame are only eligible for transmission in the corresponding departing frame. Stop-and-Go ensures that cells arrived during the same frame in the source stay in the same frame throughout the network. (r, T) traffic model is adopted in Stop-and-Go, in which a connection with a rate of $r_{i,j}$ can transmit no more than $r_{i,j} \cdot T$ bits during a frame with a length of T. If each server along the connection path guarantees that cells arrived during one frame can always be transmitted in the next frame, end-to-end delay bounds can be guaranteed.

The framing strategy introduces a coupling problem between delay bound and bandwidth granularity. Consider a framing with a length of T. The minimum bandwidth allocated to a connection is $\frac{L}{T}$, where L is the length of one cell. Therefore, a large T is preferable for a fine bandwidth allocation. However, a small T is desired to reduce delay. The coupling problem was resolved by adopting hierarchical framing strategy as described in [32].

1.4.2.2 Hierarchical Round Robin Scheduler:

Hierarchical Round Robin (HRR) adopts a hierarchical framing strategy [27]. The time axis in HRR is also segmented into frames, each of which consists of a number of slots. A slot at a higher level frame can either be assigned to a connection or to a lower level frame. The server scans through the frame in a round robin fashion. If the current slot is assigned to a connection, a packet from this connection is transmitted when there are packets waiting; otherwise, the server stays idle. Therefore, HRR is a non-work-conserving scheduler. If the current slot is assigned to a lower level frame, a slot of the lower frame is served in the same fashion.

Table 1.2 summarizes the performance of Stop-and-Go and HRR. Both Stop-and-Go and HRR can maintain smoothness in a networking environment owing to their non-work-conserving nature. In Stop-and-Go, cells arrived in the same frame at the network entrance will be transmitted in the same frame throughout the network, while in HRR cells arrived in the same frame can be transmitted in different frames.

1.5 Traffic Scheduling in Combined Input Output Queued Switches

Combined input and output queueing (CIOQ) approach was proposed to resolve the trade off between the QoS guarantees and scalability [20, 21], in which buffers are

Table 1.2 Performance of representative non-work-conserving schedulers for output-queued switches. T is the frame size and θ is the constant delay between an arriving frame and its corresponding departing frame

schedulers	traffic constrain	delay	delay jitter	buffer space
Stop-and-Go	(r, T)	$T + \theta$	T	$r(2T + \theta)$
HRR	(r, T)	$2T$	$2T$	$2rT$

placed at both sides of a switch. CIOQ emulates an output scheduling algorithm in such a way that the cell transmission order in CIOQ is the same as that in the emulated output scheduling algorithm in every time slot. Since it is only necessary to forward the cell to be transmitted in the next time slot in the emulated OQ scheduler to the output side of the switch, CIOQ does not require speedup of N to guarantee the same performance. It has been proven [20] that a switch using CIOQ with speedup of 2 can provide the same QoS guarantees as a switch using an OQ scheduling algorithm.

1.5.1 Algorithm Description

Push-in queue concept was used in [20], i.e., arriving cells can be placed at any position in the queue and once the cells are placed in the queue, the relative order cannot be changed. In a CIOQ switch, each input maintains an input queue, where the cells are stored in a specific order. Different algorithms are characterized by the orders the cells are stored in each input queue. Likewise, each output maintains an output queue which consists of cells waiting for departure. Moreover, an output priority list, which is an ordered list of cells stored in inputs and directed to that particular output, is maintained in each output. The cells on the output priority list are always ordered according to the emulated output scheduling algorithm.

A variety of CIOQ algorithms have also been proposed to emulate OQ scheduling with a speedup of 2. The algorithms differ from each other only in the insertion policy. For example, placing an arriving cell as far from the head of its input queue as possible leads to Critical Cells First (CCF) [20], and placing an arriving cell at the front of the input queue leads to Last In Highest Priority (LIHP) [20]. Each time slot is divided into four phases [20]:

1. The arrival phase: New cells arrive only in this phase.
2. The first scheduling phase: Cells are scheduled for transmissions from inputs to outputs.
3. The second scheduling phase: Again, cells are scheduled for transmissions from inputs to outputs.
4. The departure phase: Cells are transmitted out from the outputs only in this phase.

Since CIOQ requires speedup of 2, two scheduling phases are needed. During each scheduling phase, a stable matching algorithm is used to calculate a stable matching between inputs and outputs. A matching is said to be stable if for a cell c in an input, one of the following holds:

1. Cell c is part of the matching, i.e., c is going to be transmitted from the input side to the output side during this phase.
2. A cell which is ahead of c in its input queue is part of the matching.
3. A cell which is ahead of c in its output queue is part of the matching.

Note that conditions 2 and 3 could be satisfied at the same time. A stable matching algorithm given by Gale and Shapely [33] can find a stable matching within at most M iterations, where M is the total length of all the input queues. It is proven that in CIOQ, cells to be transmitted in the next time slot in the emulated output scheduler can always be forwarded to the output side, and thus the same QoS guarantees as the emulated output scheduler can be provided.

1.6 Contributions of the Dissertation

In chapter 2, we consider the scheduling problem under uniform traffic. We model and analyze the back pressure problem with independent Bernoulli traffic load, and show that back pressure occurs with high probability under loaded traffic. The average queue length at the input buffer is also derived. To address the above issues in input-queued switches, we propose a maximum size matching based algorithm, referred to as min-max fair input queueing (MFIQ), which minimizes the additional delay caused by back pressure and at the same time provides fair service among competing sessions.

As pointed out in [13, 15], maximum size matching based algorithms do not perform well for non-uniform traffic. There are several maximum weight matching based algorithms proposed in the literature [13, 16, 15] to achieve high throughput under non-uniform traffic. However, only aiming at maximizing throughput could generate adverse effects on traffic shape and quality of service (QoS) features such as delay and fairness. In chapter 3, two algorithms are proposed to provide some QoS features while achieving high throughput for input-queued switches. By setting the weight of an edge in the bipartite graph to the normalized queue length of the corresponding VOQ, the longest normalized queue first (LNQF) [18, 34] provides fairer service than LQF and better traffic shape than OCF. The stability of LNQF is also proven. In earliest due date first matching (EDDFM), the weight is a function of delay bound and thus forces the cells with earliest delay due date to have highest priority to receive service. Simulation results show that EDDFM has lower probability of delay over due than LQF, LNQF, and OCF.

The maximum matching based algorithms can only achieve asymptotical 100% throughput, and thus, cannot provide deterministic QoS guarantees. In Chapter 4, a frame based scheduling algorithm, referred to as Store-Sort-and-Forward (SSF) [19], is proposed to provide QoS guarantees for input-queued switches without requiring speedup. SSF uses a framing strategy in which the time axis is divided into constant-length frames, each made up of an integer multiple of time slots. Cells arrived during a frame are first held in the input buffers, and are then "sorted-and-transmitted" within the next frame. A bandwidth allocation strategy and a cell admission policy are adopted to regulate the traffic to conform to the (r, T) traffic model. A strict sense 100% throughput is proved to be achievable by rearranging the cell transmission orders in each input buffer, and a sorting algorithm is proposed to order the cell transmission. The delay and delay jitter are bounded by the transmission time of one frame. It is proved that a perfect matching can be achieved within $N(\ln N+O(1))$ effective moves.

Chapter 5 presents our contributions on ATM multicasting. The routing and signaling protocols for supporting multipoint-to-multipoint connections in ATM networks have been presented in recent publications. VP-Merge and VC-Merge techniques have been proposed as the likely candidates for resolving the sender identification problem associated with these connections. The additional buffer requirements in the VC-Merge mechanism and the limitations of VPI space in the VP-Merge mechanism have been the main reasons for concern about their effective utility. In this chapter, we propose improvements to these traditional merging techniques. Our proposal describes a scalable VP-Merge scheme and analyzes different mechanisms to implement the scheme in ATM networks. To facilitate an elegant implementation, we introduce VP-VC Switching, a new switching mode different from traditional VP Switching and VC Switching modes. We also propose an improved VC-Merge technique [35] to control the additional buffers at intermediate merge points. Aptly named Dynamic Multiple VC-Merge (DMVC), Fixed Multiple VC-Merge (FMVC) and Selective Multiple VC-Merge (SMVC), these mechanisms define a generic scheme for merging the data from multiple senders onto one or more outgoing links. By appropriately choosing the number of connection identifiers per connection, these schemes lead to a large reduction in the buffer requirements and an effective utilization of the VPI/VCI space. Based on extensive simulations, we show that by using two connection identifiers per connection, there is an 80% reduction in buffer requirements for DMVC and FMVC when compared to the buffer required for traditional VC-Merge.

CHAPTER 2

MIN-MAX FAIR INPUT QUEUEING (MFIQ)

It was shown [22] that the throughput of an input-queued switch is limited to 0.586 when a single first-in-first-out (FIFO) queue is used in each input port under uniform Bernoulli traffic. This is mainly owing to the head of line (HOL) blocking, i.e., if cells in the front of the input queue are blocked, the cells stored behind them cannot be transmitted even if their destination output ports are open. Since the publication of the seminal paper by Karol *et al.* [22], many works [36, 14] have indicated that the throughput of the input-buffered switch can be improved by using well designed scheduling algorithms. Parallel Iterative Matching (PIM) will reach 100% throughput if a sufficiently large number of iterations are used. Iterative round-robin matching with slip (*i*SLIP) [14] was proposed later that has similar performance as PIM at much lower hardware complexity.

While scheduling algorithms designed for output-buffered switches, like Packet Fair Queueing (PFQ), can provide end-to-end delay bound and fairness among sessions, they are not directly applicable to input-buffered switches without causing performance degradation. When the scheduler at an input port schedules one cell for transmission, schedulers at other input ports could also schedule cells for transmission to the same output port. Only one cell can get through the fabric, and the other cells are back pressured and stored in the input ports. Instead of wasting bandwidth, the scheduler in a back pressured port should schedule another cell for transmission. As a result, the back pressured sessions suffer extra delay and lose their fair share of the bandwidth while other sessions get more services than they should. The key issue is what type of actions should a scheduler executes when the output port is open for the back pressured sessions after a certain time lapse. Instead of only aiming at maximizing the throughput, we propose an algorithm, called min-max fair input queueing (MFIQ), to minimize the additional delay and to arbitrate fair service among all competing sessions in an input port.

The rest of the chapter is organized as follows. The model of the back pressure problem under independent Bernoulli traffic is presented in Section 1, and it is shown that the effect of back pressure is significant. Section 2 presents our proposed scheduler, and simulation results are shown in Section 3. Remarks are concluded in Section 4.

2.1 Queueing Analysis

Consider a system with N input ports, N output ports, and an $N \times N$ fabric. The cell arrival processes at the input ports are assumed independent and identical Bernoulli. Let p be the traffic load, i.e., in any given time slot, the probability that a cell arrives at a particular input port is p. The cell has equal probability of $1/N$ to go to any given output port, and successive cells are independent. To eliminate HOL blocking, virtual output queueing is adopted, i.e., N separated queues, each of which is associated with a corresponding output port, are maintained in an input port. The probability of a cell appearing at a virtual output queue (VOQ) is p/N. There are N^2 VOQs in the system as shown in Figure 2.1. Associated with an output port is a queue group with N VOQs, each of which is located in one of the N input ports. The cell arrival processes in these queues are also independent. Let A be the total number of cells appeared at the head of each VOQ in a queue group in one time slot. Thus, A is a Binomial random variable and has the following distribution:

$$P(A=i) \;=\; \binom{N}{i} (p/N)^i (1-p/N)^{N-i}. \qquad (2.1)$$

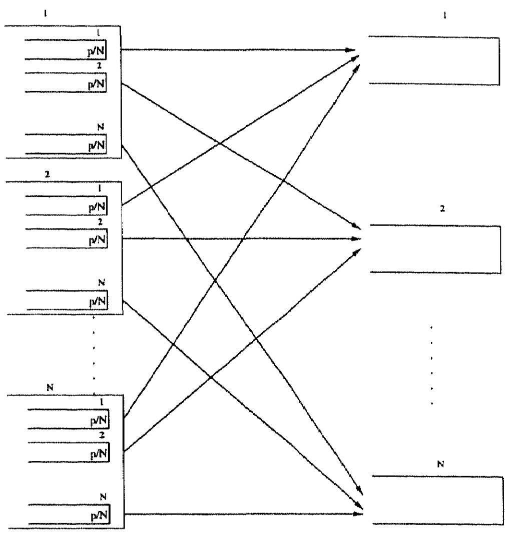


Figure 2.1 Virtual output queueing

Contention occurs when more than one of the VOQs have cells in the same time slot. Only one out of the competing cells can get through the fabric, and the others are back pressured for later transmission. Thus, back pressure occurs with probability

$$\begin{aligned} P_B &= \sum_{i=2}^{N} \binom{N}{i} (p/N)^i (1-p/N)^{N-i} \\ &= 1-(1-\frac{p}{N})^{N-1}(1-\frac{p}{N}+p). \end{aligned} \tag{2.2}$$

When the number of input ports increases, the probability of back pressure becomes

$$\lim_{N\to\infty} P_B = 1-e^{-p}(1+p). \tag{2.3}$$

Note that Equation (2.3) is monotonically increasing with the load p. The probability of back pressure reaches its maximum value of $1-2/e$ when the load reaches one.

Consider the total queue size of one queue group. With the assumption that one cell can always be transmitted during the time slot if there are cells at the head of the N VOQs, the Markov chain for the total queue length of the queue group can be obtained as shown in Figure 2.2, where

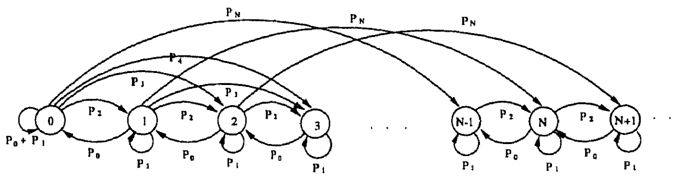


Figure 2.2 The discrete time Markov chain for the total queue length of one queue group

$$p_i = \binom{N}{i} (P/N)^i (1 - P/N)^{N-i},$$

and the state value of the chain indicates the queue size of the queue group. The transition probability matrix of the Markov chain is

$$\mathbf{P} = \begin{bmatrix} p_0 + p_1 & p_2 & p_3 & \cdots & p_N & 0 & \cdots \\ p_0 & p_1 & p_2 & \cdots & p_{N-1} & p_N & \cdots \\ 0 & p_0 & p_1 & p_2 & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}.$$

It is difficult to derive the close form distribution of the queue length, but the generating function of the queue length distribution can be readily derived:

$$Q(z) = \frac{(1-p)(z-1)}{z - A(z)}, \tag{2.4}$$

where

$$A(z) = (1 - \frac{p}{N} + z\frac{p}{N})^N \tag{2.5}$$

is the generating function of random variable A. Thus, the mean steady-state queue length can be obtained as follows:

$$\overline{Q} = \lim_{z \to 1} Q'(z) = \frac{N-1}{N} \frac{p^2}{2(1-p)}, \tag{2.6}$$

where $Q'(\cdot)$ is the derivative of $Q(\cdot)$. Figure 2.3 shows the average total queue length in a queue group associated with one output port.

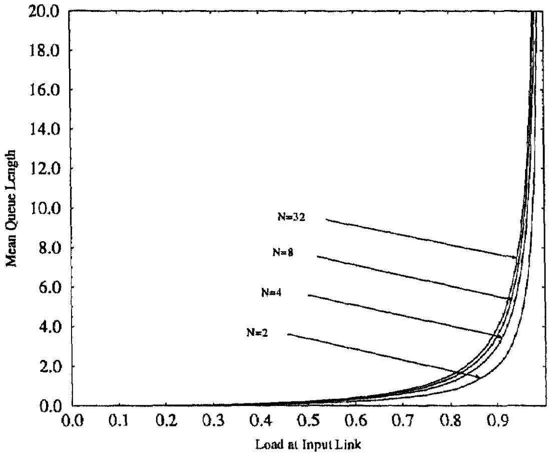


Figure 2.3 Mean queue length of a queue group with ideal throughput

Note that Equation (2.6) is applicable to every queue group. If queue groups corresponding to different output ports are independent, the average queue length at the input buffer of the switch is $N\overline{Q}$. Since the traffic in the N input ports are independent, the average queue length in one input port is also $\overline{Q}$. The Markov chain is derived based on the assumption that there is always a cell to be transmitted to a given output port whenever there are cells in the queue group. In fact, if more than one output port schedule the same input port to transmit cells, only one cell can be transmitted. As a result, the bandwidths of the other output ports are not utilized. Thus, the actual throughput can only be lower than what is assumed here. Hence, the average queue length at the input buffer in a realistic situation (Figure 2.4) is longer than the ideal case (Figure 2.3). Thus, back pressure can potentially cause adverse effect on the delay and fairness of the competing sessions.

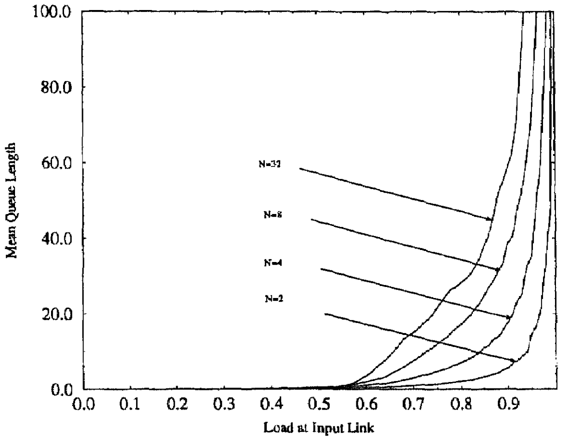


Figure 2.4 Mean queue length of a queue group simulation results

2.2 Min-max Fair Input Queueing Algorithm

Before we proceed to describe our algorithm which will overcome the shortcomings of applying PFQ directly to input-buffered switches, we first define the following.

Definition 1 *A reference scheduler(RS) of a system is an ideal scheduler which operates without back pressure but has the same configuration as the real scheduler in the system.*

The reference scheduler maintains its own virtual time. The virtual time of a session scheduled by the reference scheduler is updated no matter whether the session is back pressured in the real system. Thus, virtual time of sessions in the reference system keeps track of the service that the sessions should receive in the real system.

Definition 2 *The additional delay of a cell is the time interval between the time when the cell is transmitted in the real system and the time when the cell is scheduled in the reference scheduler.*

Note that the additional delay is negative when the cell is transmitted before it is scheduled in the reference scheduler.

Definition 3 *The normalized service lag of a session is the difference between the normalized service time the session should receive in the reference scheduler and the normalized service time it has received in the real system.*

For input-buffered switches, the schedulers of input ports are not independent from each other. When more than one input port schedules cells to the same output port, contention occurs. Only one of the competing cells can get through the fabric, and the others are back pressured in the input ports. To increase throughput, the scheduler of the back pressured input port needs to schedule another cell that is free of contention. Thus, the back pressured cell experiences additional delay and loses its fair share of service. On the contrary, the being served session receives earlier and more service than its fair share. There could be more than one session back pressured at the same time. When the output ports for the back pressured cells are open for transmission, which cell should be transmitted? Within the context of GPS which is the perfectly fair scheduler, the back pressured session with the largest normalized service lag is the one that has been back pressured longest, thus experiencing the largest additional delay. It is therefore intuitively fair to transmit the cell of the session that has been back pressured the longest. This is the essence our algorithm as shown in Figure 2.5.

In the algorithm, a reference virtual time system and a real virtual time system are maintained. The virtual time of each session is updated in the reference system, independent of the status of the real system, to keep track of the normalized service the session should receive. Normalized service lags are maintained in the real system. The system potential $V(t)$ can be updated by using any PFQ algorithms like WFQ [2], $SCFQ$ [4], WF^2Q [6], and WF^2Q+ [37]. For example, if WFQ is

```
Cell_reaches_head_of_queue(session i, packet *cell)
  if $i \in B$
    if queue_length(i) == 0
      $S_i \leftarrow max(V, S_i)$
    $F_i \leftarrow S_i + l/r_i$
  else
    $S_i \leftarrow V$
    $lag_i \leftarrow 0$
    $B \leftarrow B \cup i$
  enqueue(session i, packet *cell)

Transmit_cell
  q_to_update $\leftarrow min_i \; F_i$
  q_to_send $\leftarrow max_i \; lag_i \; \{i \mid i \in not \; BP\}$
  if q_to_send $\neq -1$
    dequeue(q_to_send)
    Update_system_potential()
    $S_{q\_to\_update} \leftarrow S_{q\_to\_update} + l/r_{q\_to\_update}$
    $F_{q\_to\_update} \leftarrow S_{q\_to\_update} + l/r_{q\_to\_update}$
    if q_to_update $\neq$ q_to_send
      normalized_lag[q_to_update]
       $\leftarrow$ normalized_lag[q_to_update] $+ \; l/r_{q\_to\_update}$
      normalized_lag[q_to_send]
       $\leftarrow$ normalized_lag[q_to_send] $- \; l/r_{q\_to\_send}$

Session_leave(session i)
  $B \leftarrow B \setminus i$
```

Figure 2.5 The pseudo-code of the min-max fair input queueing algorithm

selected to update the system potential, the rule is

$$V(t+\tau) \;=\; V(t) + \frac{\tau}{\sum_{i \in B} r_i}, \tag{2.7}$$

where B represents all backlogged sessions and τ is the time increment. The session with the smallest virtual finishing time F_i in the reference scheduler is updated regardless of the status of the real system, i.e., the virtual finishing time of the selected session i in the reference system is updated no matter whether it is back pressured or not. The session with the largest normalized lag is scheduled for transmission in the real system. If the transmitted session j is not the session selected

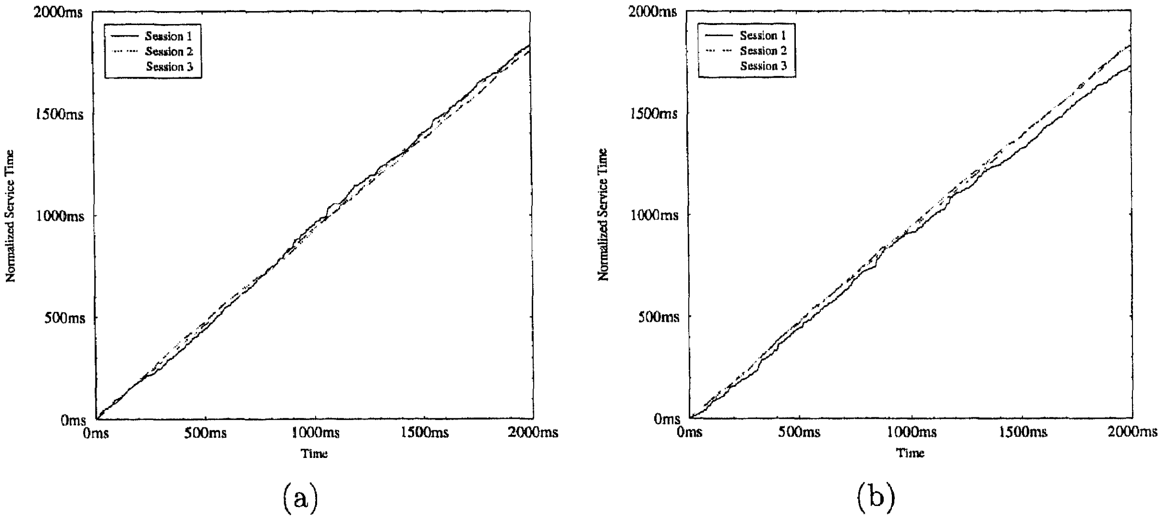


Figure 2.6 Comparison of normalized service time received by the three sessions: (a) MFIQ, and (b) reference scheduler

in the reference system, the selected session is deferred for transmission. Thus, the normalized service lag of the selected session is increased by l/r_i, and that of the transmitted session is decreased by l/r_j.

2.3 Simulation Results and Performance Comparison

Consider a system with eight input ports and eight output ports. To eliminate HOL blocking, virtual output queueing is used. Each VOQ has three sessions with transmission rates of 1, 5 and 10 Mbps. *WFQ* was selected to update the system potential. Simulations were conducted for a load of 0.8, 0.9 and 0.95. Each simulation lasted through 2 seconds.

Two schedulers were simulated: our proposed MFIQ algorithm, and the reference scheduler defined earlier, in which sessions were scheduled only based on their virtual times in the reference scheduler. When the scheduled session was back pressured, its virtual time was updated and another session which was free of

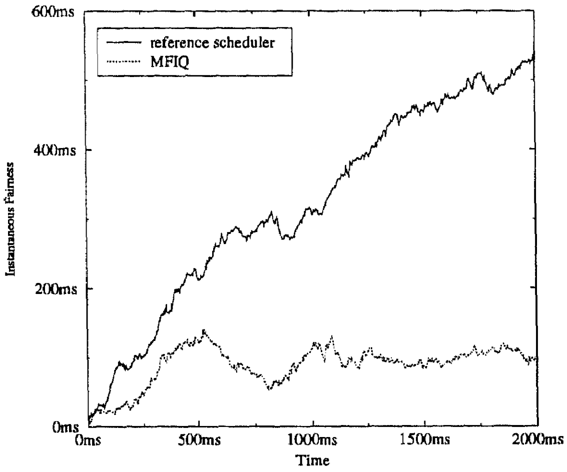


Figure 2.7 Instantaneous fairness: MFIQ versus reference scheduler

contention was selected for transmission. The scheduler did not keep track of the service time lost by the back pressured sessions. Thus, the lost service time could not be compensated.

The normalized service times received by three sessions belonging to different VOQs are shown in Figure 2.6. Since our proposed algorithm always selected the session with the largest normalized service lag for transmission, the differences of the received normalized service times among sessions were smaller than that of the reference scheduler. The instantaneous fairness, which is the difference between the largest normalized service lag and the smallest normalized service lag experienced by all sessions, is compared in Figure 2.7. The instantaneous fairness of MFIQ is much smaller than that of the reference scheduler, even though they both experience randomness owing to the back pressure. The proposed MFIQ has better performance in terms of maximum normalized service lag and maximum additional delay, as shown in Figures 2.8 and 2.9.

Table 2.1 illustrates the statistics of the simulations for different loads. Four results can be derived from the table.

Table 2.1 Statistics of the simulation results: d_i is the delay of session i and r_i is the rate of session i. F_S is the instantaneous fairness of the algorithm. D_{max} is the maximum additional delay caused by contentions while Lag_{max} is the maximum normalized service lag of the algorithm.

load	0.8		0.9		0.95	
schedulers	RS	MFIQ	RS	MFIQ	RS	MFIQ
d_1(ms): r_1=1Mbps	0.98	1.17	2.07	2.54	5.24	4.24
d_2(ms): r_2=5Mbps	0.50	0.41	0.88	0.82	1.85	1.49
d_3(ms): r_3=10Mbps	0.28	0.28	0.43	0.61	1.35	1.01
queue length (cell)	19.97	19.33	39.56	39.26	71.89	75.77
F^S(ms)	327.45	51.87	314.75	88.45	378.96	88.83
D_{max}(ms)	3.47	3.20	5.78	4.22	11.09	5.19
Lag_{max}(ms)	66.66	15.87	110.73	47.13	116.35	41.65

1. The two algorithms have similar performance in terms of average delay and average queue length.

2. The proposed MFIQ has better performance in terms of fairness and additional delay.

3. The average delay of a session is inversely proportional to its rate.

4. Average instantaneous fairness, average maximum additional delay, and average queue length increase as the load increases.

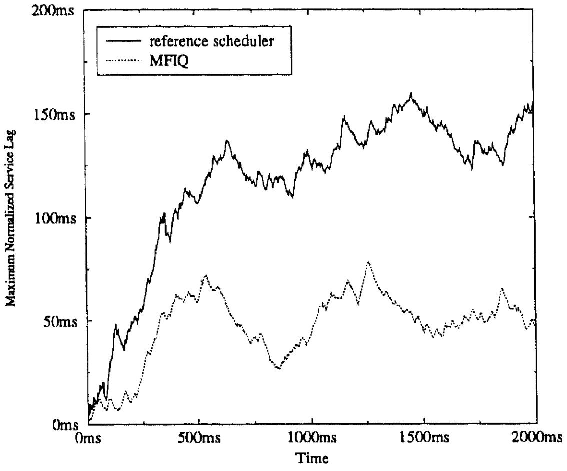


Figure 2.8 Maximum normalized service lag: MFIQ versus reference scheduler

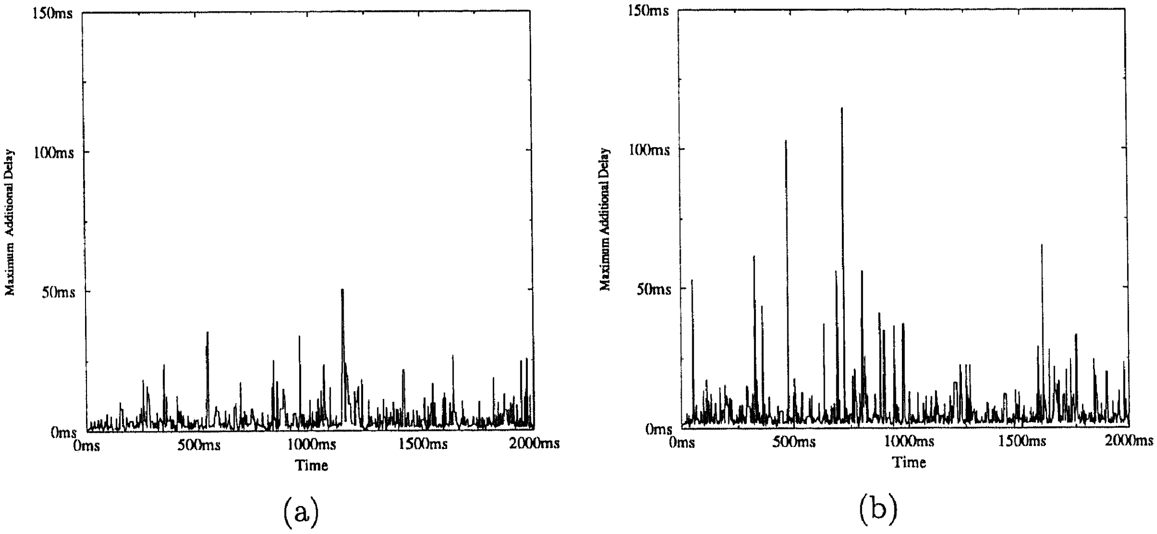


Figure 2.9 Comparison of maximum additional delay: (a) MFIQ, (b) reference scheduler

CHAPTER 3

MAXIMUM WEIGHT MATCHING

Maximum size matching based algorithms work well for uniformly distributed traffic. However, it was pointed out that a switch using maximum size matching based algorithms is not stable for non-uniform traffic [13, 15], i.e., the expected queue length in the switch could increase without bound. Round robin scheduler, which has low implementation complexity, is adopted in *i*SLIP to resolve the contention among cells stored in the same input port. However, the priority of a round robin scheduler is not a function of the queue length. Thus, *i*SLIP performs poorly for non-uniform traffic, in which the average queue length of the FIFO queues could differ strikingly under loaded traffic. Maximum weight matching can achieve high throughput under both uniform and non-uniform traffic in which each session is assigned a weight and a match with the maximum aggregate weight is obtained. Longest queue first (LQF) [13] and oldest cell first (OCF) [15] are among the maximum weight matching approach, in which the queue length and the delay time of head of line cell are set as the weights, respectively.

Algorithms which only aim at maximizing throughput could generate adverse effects on traffic shape and quality of service (QoS) features such as delay and fairness. In LQF, the priority is set according to the queue length, i.e., the queue with the largest length has the highest priority to receive service. Since the queues of the VOQs with different arrival rates are built up at different speeds, using the queue length as the weight forces the scheduler to serve the VOQs with high arrival rates and starve the VOQs with low arrival rates. This is the main reason why LQF leads to unfair service and uncontrollable delay time for the VOQs with low arrival rates. On the other hand, OCF avoids starvation by setting delay time as the weight, in which the unserved cells get growing "older" until they eventually become "old" enough to be served. Using delay time as the weight forces the scheduler to serve the VOQs burst by burst, thus sacrificing the QoS requirements. By observing that session rates and delay requirements should be incorporated in the scheduler design in order to satisfy the QoS requirements, we propose two new algorithms, referred to as longest normalized queue first (LNQF) and earliest due date first matching (EDDFM).

The rest of the chapter is organized as follows. In Section 1, we describe our switch and traffic models. Section 2 presents one of our proposed algorithms, referred to as longest normalized queue first (LNQF) and proves that a switch using LNQF is stable under both uniform and non-uniform traffic. Section 3 proposes another algorithm: earliest delay due date first matching (EDDMF) . Section 4 shows the performance of the proposed algorithm. Concluding remarks are given in Section 5.

3.1 Switch and Traffic Models

Consider an $N \times N$ input-queued ATM switch consisting of N inputs, N outputs and an $N \times N$ crossbar. To eliminate the HOL blocking, virtual output queueing is adopted, as shown in Figure 3.1.

Let $Q_{i,j}$ denote the VOQ directed to output j at input i, and $A_{i,j}$ denote the arrival process to $Q_{i,j}$. To provide QoS features, switch resources such as the bandwidth and storage should be allocated on a per-session basis. There could be more than one session arrived at a certain input directed to the same output. Thus, multiple sessions could share the same VOQ, each of which is maintained as a FIFO queue. Let $I_{i,j,k}$ be the kth session in $Q_{i,j}$ with arrival rate $\lambda_{i,j,k}$. Therefore, the arrival rate of $A_{i,j}$ can be expressed as $\lambda_{i,j} = \sum_k \lambda_{i,j,k}$. An arrival process A_i, which is the aggregate arrival process to input i, is said to be uniform if $\lambda_{i,m} = \lambda_{i,n}$, $\forall$

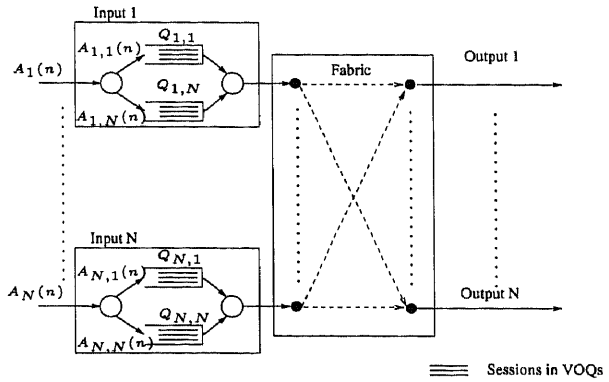


Figure 3.1 Input-queue switch model

$m \neq n$, $1 \leq m, n \leq N$. Otherwise, the process is said to be non-uniform. The traffic pattern is admissible if and only if $\sum_{j=1}^{N} \lambda_{i,j} \leq 1$ and $\sum_{i=1}^{N} \lambda_{i,j} \leq 1$.

The traffic in a real network is highly correlated from cell to cell, and cells tend to arrive at the switch in "bursts." One way of modeling a bursty source is by using an ON-OFF model in the discrete-time domain. This model is equivalent to a two-state Markov Modulated Deterministic Process (MMDP) [38]. The two states, OFF state and ON state, are shown in the Figure 3.2. 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 (P). The source can independently shift from one state to another as shown in Figure 3.2. In a discrete-time domain, state changes may occur only at the end of a time-slot. At each time slot, 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. It must be remembered 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 $P_{off} = \frac{\beta}{(\alpha+\beta)}$ and $P_{on} = \frac{\alpha}{(\alpha+\beta)}$, respectively.

The bursty source is characterized by the peak cell rate (P), the average cell rate (A), and the average number of cells per burst (B). The burstiness of the traffic

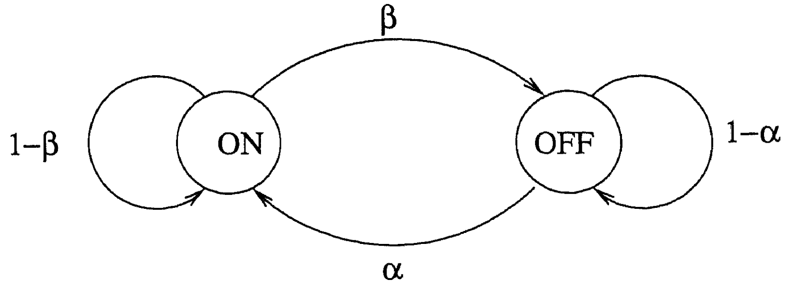


Figure 3.2 Simple ON-OFF traffic model

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 $\alpha = \frac{A}{B(P-A)}$ and $\beta = \frac{1}{B}$.

3.2 Longest Normalized Queue First (LNQF) Algorithm

The basic objective of scheduling an input-queued switch is to find a contention free match based on the connection requests. At the beginning of every time slot, each input port sends requests to the scheduler. The scheduler selects a match between the input ports and output ports with the constraints of unique pairing, i.e., at most one input port can be matched to each output port and vice versa. At the end of the time slot, a cell is transmitted per matched input-output pair.

3.2.1 Algorithm Description

Denote $l_{i,j,k}(n)$ as the length of the FIFO queue corresponding to session $I_{i,j,k}$ in $Q_{i,j}$ and $l_{i,j,}(n) = \sum_k l_{i,j,k}(n)$ denote the length of $Q_{i,j}$ at time slot n. In LNQF, the weight of a VOQ is set to the normalized queue length which is the total queue length of the VOQ divided by its rate, i.e., $w_{i,j}(n) = \frac{l_{i,j}(n)}{\lambda_{i,j}(n)}$. 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 port i and $S = [S_{i,j}(n)]$ be the service matrix which indicates the match between input and output ports.

$S_{i,j}(n)$ is set to 1 if input port i is scheduled to transmit a cell to output port j. Otherwise, $S_{i,j}(n)$ is set to 0. Let $\underline{S}_i(n) = (S_{i,1}(n), S_{i,2}(n), \ldots, S_{i,N}(n))^T$ be the service vector associated with input port i. The LNQF scheduler, as shown in Figure 3.3, performs the following for each time slot n:

1. Each input port computes the normalized queue length of each VOQ, sets it as the weight of the VOQ, 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.,

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

   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 port, and uses the service matrix $[S_{i,j}(n)]$ to configure the fabric.

3. Each input port computes the normalized queue length of each session in the matched VOQ indicated by $\underline{S}_i(n)$, and selects the session with the longest normalized queue length for transmission.

The LNQF algorithm gives preference to the VOQs with large normalized queue lengths for transmission. Note that the average queue length of a VOQ in a fair server should be proportional to its arrival rate. Using the normalized queue length as the weight forces the scheduler to serve VOQs more fairly, thus preventing VOQs with slow arrival rate from starvation. In addition, using normalized queue length as the weight allows cells arrived later to have higher weights than cells which come earlier, therefore performing burst reduction.

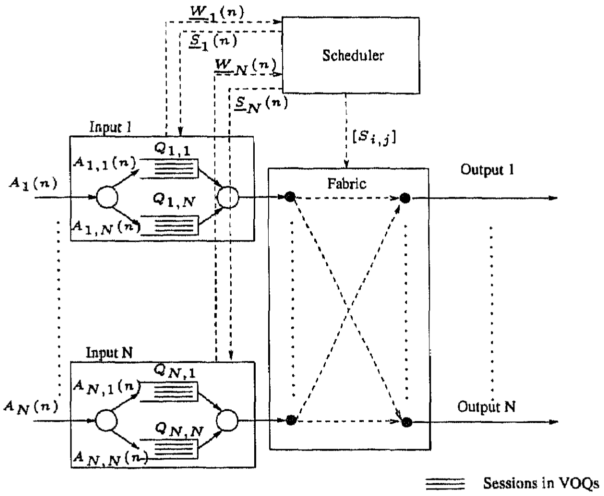


Figure 3.3 LNQF scheduler

3.2.2 Analysis of Stability

To prove that LNQF is stable, the stability of a switch is first defined.

Definition 4 *A switch is stable if and only if the expected queue length in the switch does not increase without bound, i.e.,*

$$E[\sum_{i,j} L_{i,j}(n)] < \infty, \qquad \forall n.$$

Several definitions and lemmas, similar to [13], will first be defined and proven in order to facilitate the proof of stability of LNQF.

Definition 5 *The rate matrix is defined as:*

$$\Lambda = [\lambda_{i,j}]$$

where

$$\lambda_{i,j} \geq 0, \ \sum_{i=1}^{N} \lambda_{i,j} \leq 1, \ \sum_{j=1}^{N} \lambda_{i,j} \leq 1.$$

Definition 6 *The rate vector associated with the rate matrix* Λ *is defined as:*

$$\underline{\lambda} = (\lambda_{1,1}, \lambda_{1,2}, \ldots, \lambda_{1,N}, \ldots, \lambda_{N,1}, \ldots \lambda_{N,N})^T$$

Definition 7 *The arrival vector representing the arrivals to the VOQs is defined as:*

$$\underline{A}(n) = (A_{1,1}(n), A_{1,2}(n), \ldots, A_{1,N}(n), \ldots, A_{N,1}(n), \ldots A_{N,N}(n))^T$$

where $A_{i,j}(n)$ represents the number of cells arrived at the $Q_{i,j}$ at time n.

Definition 8 *The service matrix indicating the match between inputs and outputs is defined as:*

$$S(n) = [S_{i,j}(n)],$$

where

$$S_{i,j}(n) = \begin{cases} 1 & \text{if } Q_{i,j} \text{ is selected for service at time } n, \\ 0 & \text{if } Q_{i,j} \text{ is not selected for service at time } n. \end{cases}$$

Since $\sum_{i=1}^{N} S_{i,j}(n) = \sum_{j=1}^{N} S_{i,j}(n) = 1$, the service matrix is a permutation matrix.

Definition 9 *The service vector corresponding to the service matrix is defined as:*

$$\underline{S}(n) = (S_{1,1}(n), S_{1,2}(n), \ldots, S_{1,N}(n), \ldots, S_{N,1}(n), \ldots S_{N,N}(n))^T$$

Definition 10 *The queue length vector representing the queue length of the VOQs at time n is defined as:*

$$\underline{L}(n) = (L_{1,1}(n), L_{1,2}(n), \ldots, L_{1,N}(n), \ldots, L_{N,1}(n), \ldots L_{N,N}(n))^T$$

where $L_{i,j}(n)$ represents the queue length of the $Q_{i,j}$ at time n.

Definition 11 *The normalization matrix R is defined as:*

$$R = diag[\lambda_{1,1}^{-1}, \ldots, \lambda_{1,N}^{-1}, \ldots, \lambda_{N,1}^{-1}, \ldots, \lambda_{N,N}^{-1}]$$

Definition 12 *The approximate next state vector of queue length is defined as:*

$$\hat{\underline{L}}_{i,j} = (L_{1,1}(n+1), \ldots, L_{1,N}(n+1), \ldots, L_{N,1}(n+1), \ldots, L_{N,N}(n+1))^T$$

where

$$\hat{L}_{i,j}(n+1) = L_{i,j}(n) - S_{i,j}(n) + A_{i,j}(n)$$

$\hat{L}_{i,j}(n+1)$ approximates the exact next state queue length of $Q_{i,j}$,

$$L_{i,j}(n+1) = [L_{i,j}(n) - S_{i,j}(n)]^+ + A_{i,j}(n)$$

Fact 1 *(Birkhoff's Theorem) [39] The doubly sub-stochastic $N \times N$ square matrices form a convex set, C, with the set of extreme points equal to permutation matrices.*

Lemma 1 $\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) \leq 0, \forall(\underline{L}(n), R)$, *where $\underline{S}^*(n)$ is the match (solution) with the maximum weight.*

Proof: Consider a linear programming problem as follows:

$$\max_{\underline{\lambda}}(\underline{L}^T(n)R\underline{\lambda}) \quad s.t. \ \lambda_{i,j} \geq 0, \ \sum_{i=1}^{N} \lambda_{i,j} \leq 1, \ \sum_{j=1}^{N} \lambda_{i,j} \leq 1$$

From **Fact 1** we know that doubly sub-stochastic matrices Λ forms a convex set, which has extreme points indicated by permutation matrices. The above linear programming problem has a solution at the extreme points of the convex set. Therefore,

$$\begin{aligned} \max(\underline{L}^T(n)R\underline{\lambda}) &\leq \max(\underline{L}^T(n)R\underline{S}(n)) \\ &= \underline{L}^T(n)R\underline{S}^*(n) \end{aligned} \tag{3.1}$$

Thus,

$$\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) \leq 0 \tag{3.2}$$

Lemma 2 *$E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \leq M + L$ where M and L are positive constants.*

Proof:

$$\begin{aligned}
& \hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \\
= \; & [\underline{L}(n) + \underline{A}(n) - \underline{S}(n)]^T R[\underline{L}(n) + \underline{A}(n) - \underline{S}(n)] - \underline{L}^T(n)R\underline{L}(n) \\
= \; & 2\underline{L}^T(n)R[\underline{A}(n) - \underline{S}(n)] + \underline{A}^T(n)R\underline{A}(n) - 2\underline{A}^T(n)R\underline{S}(n) + \underline{S}^T(n)R\underline{S}(n)
\end{aligned} \tag{3.3}$$

After taking expectation of Equation (3.3),

$$\begin{aligned}
& E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\
= \; & 2\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) + \underline{\lambda}^T R\underline{\lambda} + \underline{S}^{*T}(n)R\underline{S}^*(n) - 2\underline{\lambda}^T R\underline{S}^*(n) \\
= \; & 2\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) + \sum_{i,j}\lambda_{i,j} + \sum_{i,j}\frac{s_{i,j}(n)}{\lambda_{i,j}} - 2\sum_{i,j}S_{i,j}(n) \\
\leq \; & 2\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) + M + L \qquad (3.4) \\
\leq \; & M + L
\end{aligned}$$

Thus, $E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \leq M + L$, where $M = \sum_{i,j}\lambda_{i,j} \geq 0$ and $L = \sum_{i,j}\frac{s_{i,j}(n)}{\lambda_{i,j}} \geq 0$.

Lemma 3 *$E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \leq -\varepsilon \parallel \underline{L}(n) \parallel + M + L$ where ε, M and L are positive constants.*

Proof: For any rate vector $\underline{\lambda}$, a vector $\underline{\lambda}^u$ can be found to satisfy the following conditions:

$$(1-\beta)\underline{\lambda}^u = \underline{\lambda}$$

$$\sum_{i=1}^{N}\lambda_{i,j}^u \leq 1, \quad \sum_{j=1}^{N}\lambda_{i,j}^u \leq 1, \quad \lambda_{i,j}^u \geq 0, \quad \forall i,j$$

where $0 \leq \beta \leq 1$

Thus, the first term of equation (3.4) can be expressed as:

$$\begin{aligned}
& \underline{L}^T(n)R(\underline{\lambda} - \underline{S}^*(n)) \\
= \;& \underline{L}^T(n)R((1-\beta)\underline{\lambda}^u - \underline{S}^*(n)) \\
= \;& \underline{L}^T(n)R(\underline{\lambda}^u - \underline{S}^*(n)) - \underline{L}^T(n)R(\beta\underline{\lambda}^u) \\
\leq \;& -\beta\underline{L}^T(n)R\underline{\lambda}^u \\
= \;& -\frac{\beta}{1-\beta}\underline{L}^T(n)R\underline{\lambda} \\
= \;& -\frac{\beta}{1-\beta} \parallel \underline{L}(n) \parallel\parallel \underline{1} \parallel \cos\theta \\
= \;& -\frac{N\beta}{1-\beta} \parallel \underline{L}(n) \parallel \cos\theta
\end{aligned}$$

Since $L_{i,j}(n) \geq 0, \forall i, j$

$$\cos\theta = \frac{\underline{L}(n)\underline{1}}{\parallel \underline{L}(n) \parallel\parallel \underline{1} \parallel} \geq 0$$

From equation (3.4),

$$\begin{aligned}
& E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\
\leq \;& -\frac{2N\beta}{1-\beta}\cos\theta \parallel \underline{L}(n) \parallel + M + L \qquad (3.5)
\end{aligned}$$

Let $\varepsilon = \frac{2N\beta}{1-\beta}\cos\theta$, Lemma 3 is proved.

Lemma 4 $E[\underline{L}^T(n+1)R\underline{L}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \leq -\varepsilon \parallel \underline{L}(n) \parallel + M + L + \frac{N}{\lambda_{min}}$ *where* ε, M, L *and* N *are positive constants.*

Proof: The exact next state queue length is:

$$L_{i,j}(n+1) \;=\; [L_{i,j}(n) - S_{i,j}(n)]^+ + A_{i,j}(n), \qquad (3.6)$$

where $[x]^+ = \max\{0, x\}$.

From Definition (12), the approximation of (3.6) becomes

$$\hat{L}_{i,j}(n+1) = L_{i,j}(n) - S_{i,j}(n) + A_{i,j}(n) \tag{3.7}$$

Since $S_{i,j}(n)$ is either 0 or 1, the approximated next state queue length has the following relation with the exact next state queue length:

$$L_{i,j}(n+1) \quad = \quad \begin{cases} \hat{L}_{i,j}(n+1)+1 & if \;\; L_{i,j}(n)=0 \;\; and \;\; S_{i,j}(n)=1; \\ \hat{L}_{i,j}(n+1) & otherwise \end{cases}$$

Thus,

$$\underline{L}^T(n+1)R\underline{L}(n+1) - \hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) \;\; \leq \;\; \frac{N}{\lambda_{min}} \tag{3.8}$$

where $\lambda_{min} = \min\{\lambda_{i,j}\}$. From Lemma 3,

$$\begin{aligned} & E[\underline{L}^T(n+1)R\underline{L}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\ \leq \;\; & -\varepsilon \parallel \underline{L}(n) \parallel + \frac{N}{\lambda_{min}} + M + L \end{aligned} \tag{3.9}$$

Lemma 5 *There exists a quadratic Lyapunov function $V(\underline{L}(n))$, such that*

$$E[V(\underline{L}(n+1)) - V(\underline{L}(n)) \mid \underline{L}(n)] \leq -\varepsilon \parallel \underline{L}(n) \parallel + K$$

where ε and K are positive constant.

Proof: Lemma 5 follows from Lemma 4 by letting $V(\underline{L}(n)) = \underline{L}^T(n)R\underline{L}(n)$ and $K = M + L + \frac{N}{\lambda_{min}}$.

Theorem 1 *The switch using LNQF is stable for all admissible traffic patterns.*

Proof: From Lemma 5, the quadratic Lyapunov function of the queue length vector has a negative drift. According to [40], Theorem 1 is proved.

3.3 Earliest Due Date First Matching (EDDFM) Algorithm

When a cell belonging to session $I_{i,j,k}$ in $Q_{i,j}$ arrives at time slot n, the EDDFM algorithm sets the initial weight of the cell to $P_{i,j,k}(n) = P_m - DB_{i,j,k}$, where $DB_{i,j,k}$ is the delay bound of session $I_{i,j,k}$ and P_m is an integer that is greater than $max_{\forall i,j,k}(DB_{i,j,k})$. Then the cell is inserted in the queue at the position closest possible to the head of queue such that all the cells beyond this cell have smaller weights. If a cell in the queue is served in a time slot, it will be deleted from the queue; otherwise, its weight will increase by one. The weight of VOQ $Q_{i,j}$ at time slot n, $w_{i,j}(n)$, is set to the weight of the head cell of this queue if the queue is not empty, and 0, otherwise.

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 $S = [S_{i,j}(n)]$ be the service matrix which indicates the match between input and output ports. $S_{i,j}(n)$ is set to 1 if input i is scheduled to transmit a cell to output j. Otherwise, $S_{i,j}(n)$ is set to 0. Let $\underline{S}_i(n) = (S_{i,1}(n), S_{i,2}(n), \ldots, S_{i,N}(n))^T$ be the service vector associated with input i. Like LNQF as shown in Figure 3.3, the EDDFM scheduler performs the following for each time slot n:

1. Each input i set the weight of every VOQ $Q_{i,j}$ to the weight of the head cell if it is not empty, and 0, otherwise; then input i 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.,

$$\arg\max_S [\sum_{i,j} S_{i,j}(n) w_{i,j}(n)]$$

   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 service matrix $[S_{i,j}(n)]$ to configure the fabric.

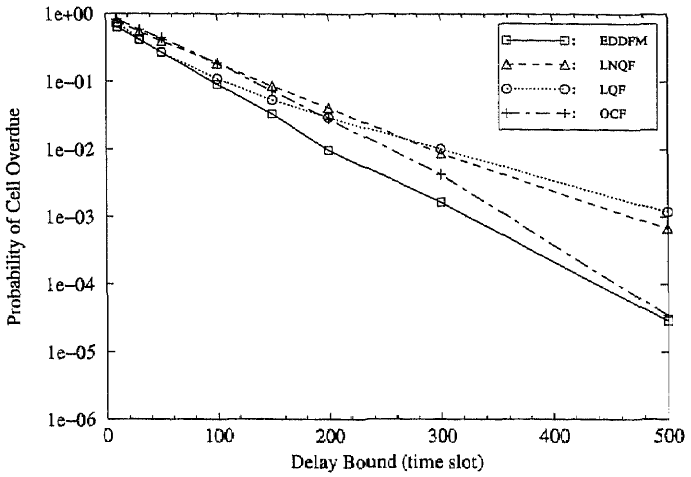


Figure 3.4 Comparison of probability of cell overdue

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

3.4 Performance Comparison of Proposed Algorithms

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-uniform, 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 were assigned according to the following rules: sessions with rates over 10% of the link capacity were treated as fast sessions and were assigned short delay, sessions with rates between 1% and 10% of the link capacity were

Table 3.1 Statistics of the simulation results: $d_{i,j}$ is the average delay of the jth session in VOQ $(1, i)$.

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

treated as medium sessions and were assigned short and medium delay randomly, and sessions with rates less than 1% of the link capacity were treated as slow sessions and were assigned short, medium and long delay randomly. The medium delay and long delay were set to five times and ten times of short delay, respectively. The configuration of delay bounds of each session remained the same for different algorithms for comparison. The probabilities of cell overdue for different algorithms are shown in Figure 3.4. The values of the delay bound in the figure are associated with short delay.

Table 3.1 summarizes the performance comparison among EDDMF, LNQF, LQF and OCF. The following results can be derived from Table 3.4 and Figure 3.4.

- Both LNQF and EDDFM are stable.
- Both LNQF and EDDFM provide comparable delay for each session, implying that they are non-starvation algorithms.

- The fairness of LNQF and EDDFM are in the same order as OCF, and are smaller than that of LQF.

- EDDFM has the lowest probability of cell overdue.

- LNQF performs well in reducing the burstiness, while EDDFM does not.

CHAPTER 4

STORE-SORT-AND-FORWARD (SSF)

There has been a trade off between QoS guarantees and scalability: the input queueing architecture is scalable but cannot provide guaranteed QoS, while the output queueing architecture can provide guaranteed QoS but is not scalable. Lately, there is a trend to adopt combined input output queueing (CIOQ), in which buffers are placed at both the input and output sides of a switch. It has been shown that a CIOQ switch with moderate speedup can be constructed to behave identically to an output-queued switch [20, 21, 41]. An algorithm called most urgent cell first algorithm (MUCFA) [41] requires a speedup of 4 in order to enable a CIOQ switch to exactly emulate an output-queued switch employing FIFO discipline. It has been proved later in [20] that a speedup of 2 is sufficient for a CIOQ switch to behave identically to an output-queued switch which employs work-conserving and monotonic scheduling discipline.

With the same buffer access and fabric speed, a switch which does not require speedup can provide N times the capacity of a switch which requires a speedup of N. Thus, speedup should be kept as low as possible, and can only be eliminated by using solely input queueing. In this chapter, an input scheduling algorithm, referred to as Store-Sort-and-Forward (SSF), is proposed and proved to be able to provide guaranteed end-to-end delay and delay jitter bounds, and to achieve strict sense 100% throughput with no speedup. As opposed to existing input scheduling algorithms [12, 13, 14, 15] which employ the work-conserving discipline, SSF uses a framing strategy, which is non-work-conserving. A switch implementing the non-work-conserving discipline may be idle even when there are cells waiting for service, thus possibly increasing the average delay. However, the end-to-end delay bound is a more important performance index than average delay for guaranteed services [9]. In the existing algorithms, cells are immediately eligible for transmission upon their arrivals to a switch, and thus the existence of a perfect matching in which every input is matched to a unique output cannot be ensured in every time slot. As a result, although 100% throughput can be achieved asymptotically, no QoS guarantees can be provided. In SSF, the time axis is divided into constant periods of length T, called frames, each of which is an integer multiple of the cell transmission time. Cells arrived at the inputs of a switch during one frame are first held in the input buffers, and are then "sorted-and-transmitted" in the next frame. The (r, T) traffic model is adopted in SSF, in which a connection with a rate of r cannot transmit more than $r \cdot T$ bits during time T. Bandwidth allocation is performed before a connection is established to assign the connection rate in such a way that the aggregate rate of any link is below its capacity. The SSF algorithm consists of a cell admission policy to regulate the traffic pattern to conform to the (r, T) traffic model at the source node of each connection, and a sorting algorithm at each switching node to resolve the input and output contentions. It is proved that cells arrived during one frame can be transmitted in the next frame. Therefore, an input-queued switch employing SSF can achieve strict sense 100% throughput, and provide deterministic end-to-end delay and delay jitter bounds. Since SSF is a non-work-conserving scheduler, the performance analysis can be extended from a single node to a network with arbitrary topology, and more efficient usage of buffer space than work-conserving schedulers can be achieved [9].

The rest of the chapter is organized as follows. Section 1 presents the SSF algorithm. The proof of the QoS guarantees and the analysis of the complexity of SSF are given in Section 2. Concluding remarks are given in Section 3.

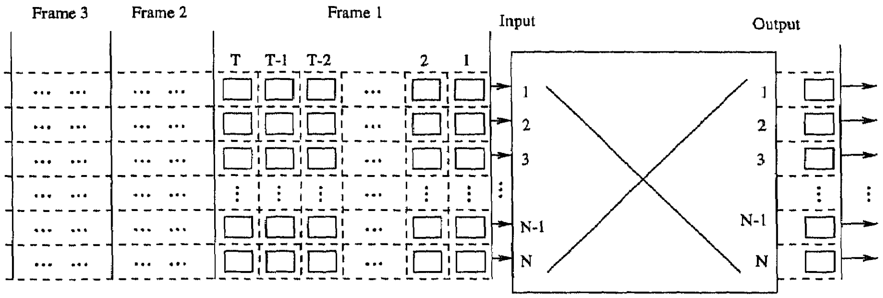


Figure 4.1 An $N \times N$ input-queued switch with time axis divided into frames

4.1 The Store-Sort-and-Forward Algorithm

The Store-Sort-and-Forward (SSF) algorithm consists of two parts: a cell admission policy which is only needed at the source node of each connection to regulate the traffic to conform to the (r, T) traffic model [7], and a sorting algorithm, which is needed at each switching node to resolve input and output contentions.

4.1.1 Framing Strategy and Cell Admission Policy

Consider an $N \times N$ input-queued cell switch in which buffers are only placed at the input side of the switch. A framing strategy similar to [7] is adopted in SSF. At each switch, the time axis is divided into frames with equal length of T, where T is an integer multiple of the transmission time of a cell. We assume that all input and output links have the same transmission rate R and are synchronized, i.e., input and output links start service at the same point of time, as shown in Figure 4.1. For simplicity, the frame size M is expressed in unit of cells, i.e., $M = \frac{1}{L} \cdot R \cdot T$, where L is the cell length in bits.

Cells arrived during one frame are first held in the input buffers and eligible for transmission in the next frame, as shown in Figure 4.2. Cells stored in one input buffer may go to any of the outputs. Let $r_{i,j}$ be the rate of a connection between input i and output j. The traffic load of any input or output link should be kept below its

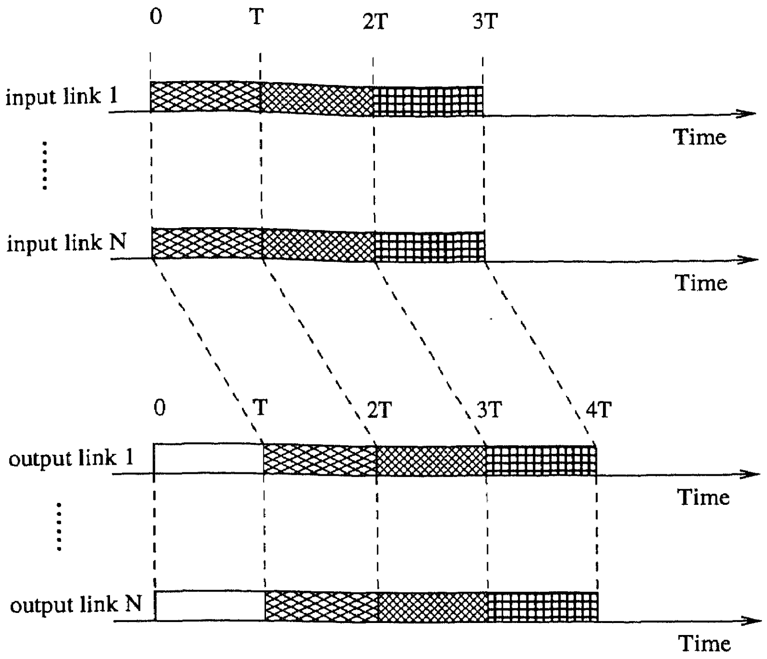


Figure 4.2 Time relation between cell arrivals and departures

capacity to avoid cell loss, i.e., $\sum_{i=1}^{N} r_{i,j} \cdot T \leq R \cdot T$ and $\sum_{j=1}^{N} r_{i,j} \cdot T \leq R \cdot T$. Thus, bandwidth allocation must be performed in the signaling phase before a connection is established. To guarantee that the traffic on any link is not overloaded, a cell admission policy is needed to regulate the traffic to conform to (r, T) traffic model, in which a connection with a rate of $r_{i,j}$ can transmit no more than $r_{i,j} \cdot T$ bits during a frame with length of T. If each server along a connection path guarantees that cells arrived during one frame can always be transmitted in the next frame, the connection will conform to the (r, T) traffic model at every switch throughout the network [7], and therefore, it is only necessary to have the cell admission function at the source node of the connection.

4.1.2 The Sorting Algorithm

The sorting algorithm[1] is a key element of SSF to ensure that cells arrived in one frame can be transmitted in the next frame by completely resolving the input and output contentions. Cells arrived at one input during a frame can go to any of the outputs. Since there is no speedup at either the buffers or the fabric, only one cell can be transmitted from each input, and likewise only one cell can be forwarded to each output in any given time slot. Thus, contentions occur when more than one cell is destining for the same output in the same time slot, as shown in Figure 4.3(a). Contention is the main problem which restricts an input-queued switch from providing QoS guarantees, and can be resolved by rearranging the transmission order of the cells among different connections[2] arrived during one frame in each input buffer in such a way that cells to be transmitted in the same time slot are going to different outputs, as shown in Figure 4.3(b). Let $C_{i,j}$ be the number of cells which arrive at input i and are directed to output j in one frame. We will prove in Section 2 that such a rearrangement always exists if the traffic pattern conforms to the (r, T) model and the traffic is not overloaded, i.e., $\sum_i C_{i,j} \leq M$ and $\sum C_{i,j} \leq M$. For simplicity, we make the following assumption.

Assumption 1 [3] *All the input and output links are fully utilized, i.e.,*

$$\sum_i C_{i,j} = M \quad \textit{and} \quad \sum_j C_{i,j} = M \quad \forall i, j = 1, 2, \cdots, N.$$

[1] F. Neri of Politecnico di Torino pointed out that the algorithm may be mapped into a similar procedure in routing CLOS networks.

[2] Changing the relative transmission order of cells belonging to an individual connection does not affect the contentions.

[3] If $\sum_i C_{i,j} < M$ or $\sum_j C_{i,j} < M$, void cells can be easily inserted into corresponding input buffers at the end of each frame to meet Assumption 1. Void cells are sorted as normal cells, but are not transmitted.

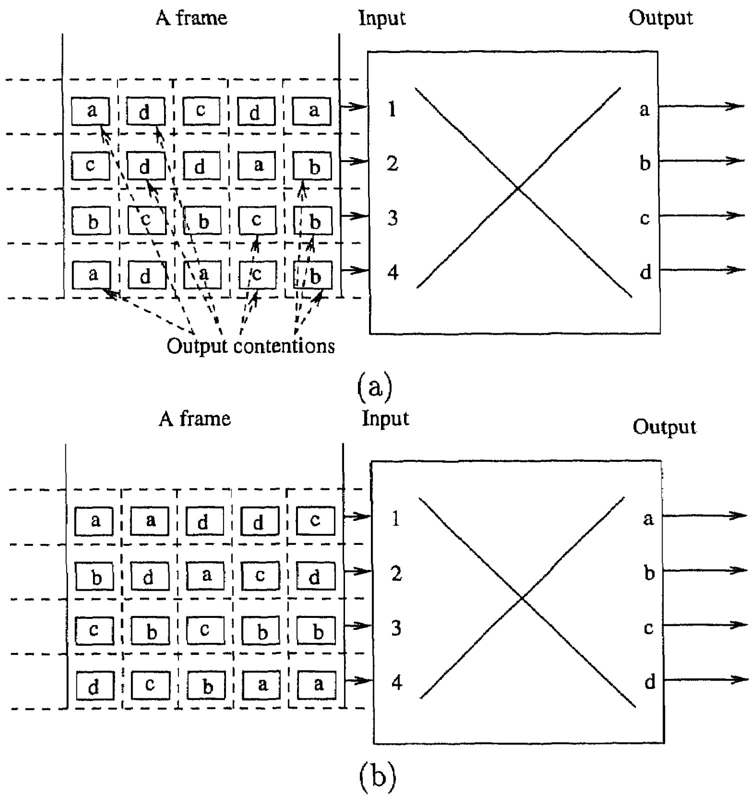


Figure 4.3 Cells arrived in one frame: (a) original arrival orders. (b) scheduled transmission orders.

Definition 13 *A perfect matching is a matching in which every input is matched to a unique output.*

To ensure that cells arrived in one frame can be forwarded to the output links in the next frame, a perfect matching is needed in each time slot. A traffic matrix $W = [w_{i,j}]$ can be generated for each frame, where every row i in W represents a distinct input k, and every column j represents a distinct output l. Note that i is not necessarily equal to k; likewise, j may not equal to l. The element $w_{i,j}$ at the intersection of row i and column j indicates the number of cells destined for the corresponding output l from the corresponding input k. The sum of elements along any row or column is exactly M based on Assumption 1. If elements along the diagonal of the matrix are all nonzero, every input is matched to a unique output. Such a matrix, referred to as a matched matrix, is attainable by swapping rows and columns of an arbitrary traffic matrix if Assumption 1 is held. A modified version of McWorter's algorithm which was originally proposed to resolve the marriage problem [42] is used to obtain a matched matrix.

Consider a traffic matrix which can be decomposed into the following form:

$$W_0 = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

where blocks A and D are square, and all diagonal elements of A are nonzero[4].

Definition 14 *An effective move consists of row and column swappings with the constraint that all diagonal elements of A remain nonzero that will result in*

- *moving a nonzero element in D to the upper-left corner of D,*
- *replacing a zero element in B by a nonzero element in A, or*
- *replacing a zero element in D by a nonzero element in C.*

A matched matrix can thus be obtained by the following iterative procedure in each time slot:

1. **Single effective move:** If there is at least one nonzero element in block D, that element can be moved to the upper-left corner of D by swapping rows and columns without changing any element in block A, as shown in Figure 4.4. Therefore, the size of A is increased by one[5] within a single effective move. This step is repeated until all elements in D become zero or the size of D becomes zero.

[4] Initially, the size of A can be 0.

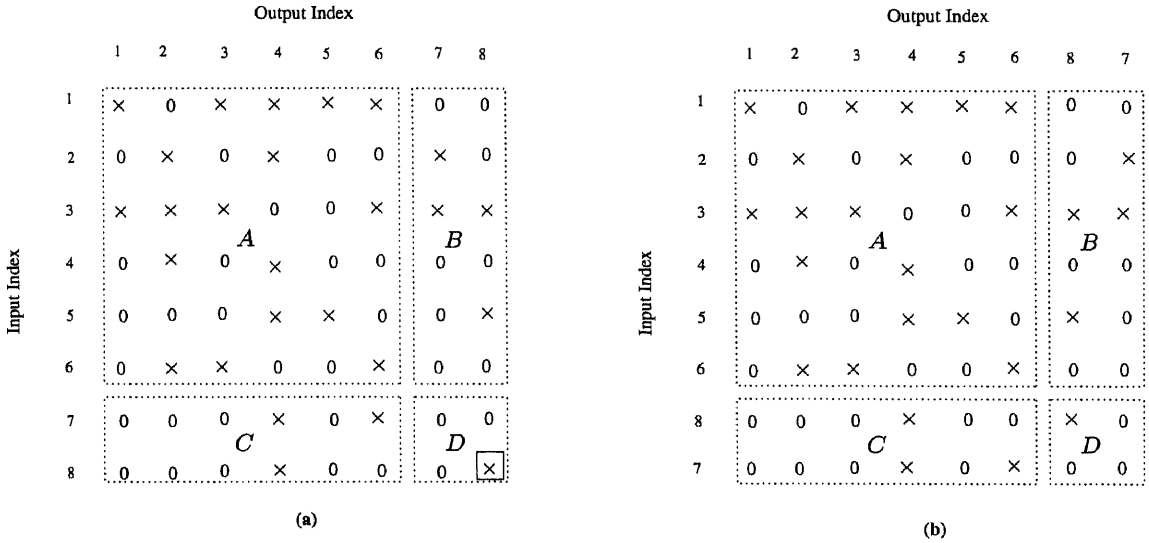
[5] We adopt the notation that the size of an $N \times N$ matrix when increased by one becomes an $(N+1) \times (N+1)$ matrix.

2. **Double effective move:** If all elements in D become zero, find a nonzero element in C. There must be some nonzero elements in block C, otherwise the input buffers corresponding to the rows in blocks C and D do not contain any cell, thus violating Assumption 1. For any nonzero element $w_{i,j}$ in block C, if there exists a corresponding nonzero element $w_{j,k}$ for some k in block B, interchange columns j and k, thus resulting in one nonzero element in block D, as shown in Figure 4.5. By repeating Step 1, the size of A is increased by one. That is, it takes two effective moves to increase the size of A by 1, thus so called a **double effective move**.

3. **Multiple effective move:** For every element $w_{i,j}$ in block C, if the corresponding element $w_{j,k}$ is zero for all k in block B, rows and columns are swapped (see Figure 4.6(b)) such that nonzero columns of C are moved right next to block D resulting in the following matrix:

$$W_1 = \begin{bmatrix} A_1 & X & G \\ F & A_2 & 0 \\ 0 & E & D \end{bmatrix},$$

where block E contains all the nonzero columns of C and all diagonal elements of A_1 and A_2 are still nonzero. Block F is called a residue matrix. We will prove in Section 2 that elements in block F cannot be all zero. For any nonzero element $w_{i,j}$ in F, if there exists a corresponding nonzero element $w_{j,k}$ for some k in G, interchange columns j and k, thus resulting in a nonzero element in the zero block just below G (i.e., completion of one effective move), as shown in Figure 4.6(c). Then, Step 2 can be repeated to augment the size of A by one. However, for every nonzero element $w_{i,j}$ in F, if the corresponding element $w_{j,k}$ is zero for all k in block G, the sub-matrix consisting of all but the extended



(a)

Input Index \ Output Index	1	2	3	4	5	6	7	8
1	×	0	×	×	×	×	0	0
2	0	×	0	×	0	0	×	0
3	×	×	×	0	0	×	×	×
4	0	×	0	×	0	0	0	0
5	0	0	0	×	×	0	0	×
6	0	×	×	0	0	×	0	0
7	0	0	0	×	0	×	0	0
8	0	0	0	×	0	0	0	☒

(b)

Input Index \ Output Index	1	2	3	4	5	6	8	7
1	×	0	×	×	×	×	0	0
2	0	×	0	×	0	0	0	×
3	×	×	×	0	0	×	×	×
4	0	×	0	×	0	0	0	0
5	0	0	0	×	×	0	×	0
6	0	×	×	0	0	×	0	0
8	0	0	0	×	0	0	×	0
7	0	0	0	×	0	×	0	0

Figure 4.4 A single effective move: (a) There is a nonzero element in block D. (b) The nonzero element is moved to the upper-left corner of block D by swapping rows 7 and 8, and then columns 7 and 8.

rows and extended columns[6] of block E satisfies the same conditions as the original situation in Step 3, and can thus be further decomposed as above. Hence, either a nonzero element can be moved to the zero block below G or the traffic matrix can be recursively decomposed.

We will prove in Section 2 that a perfect matching can be found within $N(\ln N + O(1))$ effective moves if the traffic pattern conforms to Assumption 1. Once a matched matrix is found in a time slot, a cell corresponding to each diagonal elements, say $w_{i,i}$, is scheduled to be transmitted during the next frame from the input corresponding to row i to the output corresponding to column i. Then the value of each diagonal element is reduced by one. In the next time slot, a new matched matrix

[6] An extended row of a block is referred to as the row of the traffic matrix which contains the row of that block; an extended column of a block is referred to as the column of the traffic matrix which contains the column of that block.

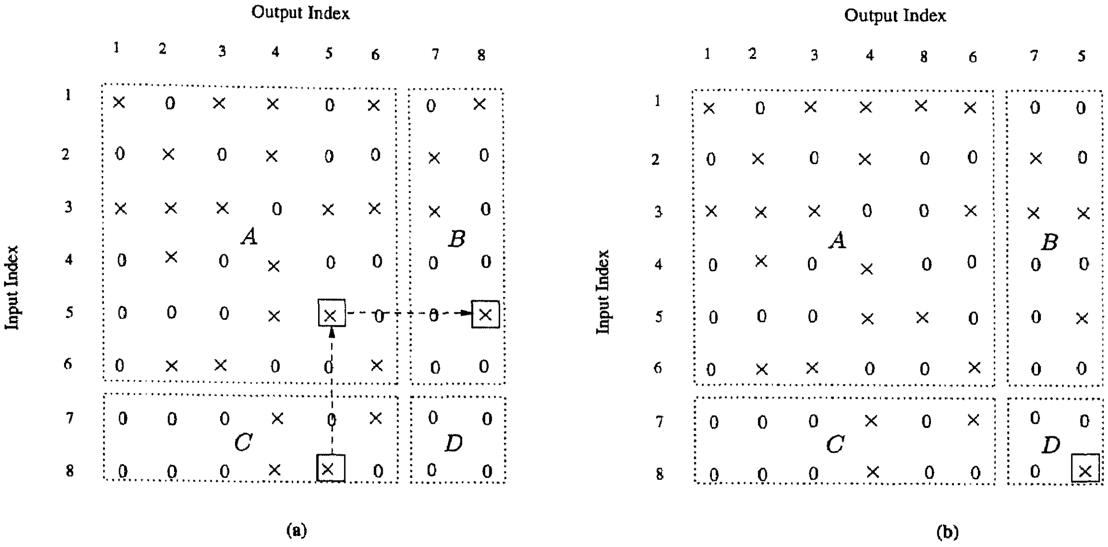


Figure 4.5 A double effective move: (a) Block D has only zeros and there is a nonzero element in block B corresponding to a nonzero element in block C. (b) The nonzero element in block C is moved to block D by swapping columns 5 and 8.

is obtained based on the updated traffic matrix until all the cells in the frame are scheduled.

4.2 Algorithm Analysis

In this section, we first prove that by rearranging the cell transmission orders in each input buffer, a contention free schedule can be found in each time slot if the traffic pattern follows Assumption 1, and then we analyze the complexity of the algorithm.

4.2.1 Guaranteed QoS

Finding a perfect matching is the same as finding a system of distinct representatives (SDR) in a family of sets [43].

Definition 15 *Suppose $\Omega = \{S_1, S_2, \cdots, S_N\}$ is a family of sets, where S_i, $i = 1, 2, \cdots, N$, is a set of elements. Sets S_i and S_j are not necessarily distinct $\forall i \neq j$. Let $\underline{V} = (a_1, a_2, \cdots, a_N)$ be an N-tuple vector with $a_1 \in S_1$, $a_2 \in S_2$, $\cdots$, $a_N \in S_N$.*

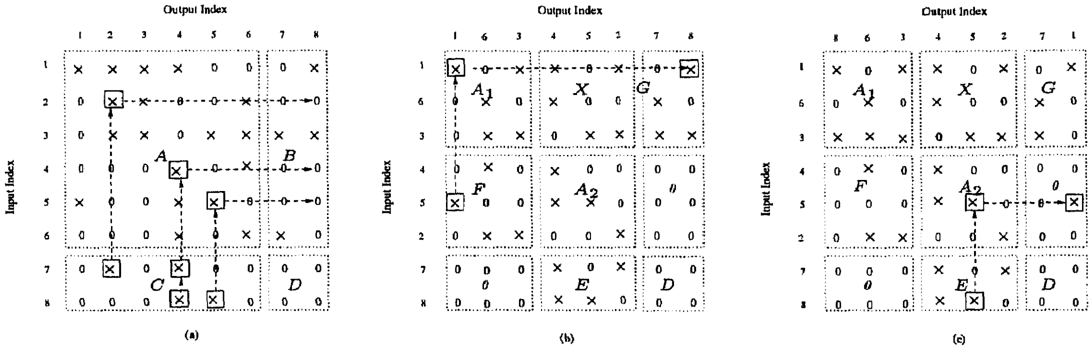


Figure 4.6 A multiple effective move: (a) There are only zero elements in block B corresponding to the nonzero elements in block C. (b) Residue matrix F is constructed by swapping columns 2 and 6, and then rows 2 and 6. There is a nonzero element in G corresponding to a nonzero element in F. (c) The nonzero element in F can be moved to the zero block below G by swapping columns 1 and 8.

Then, $\underline{V}$ is called a system of representatives for Ω. In addition, if a_i's are all distinct, $\underline{V}$ is called a system of distinct representatives (SDR) for Ω.

Fact 2 *Philip Hall's Theorem [43]: The family of sets $\Omega = \{S_1, S_2, \cdots, S_N\}$ possesses an SDR iff for all $k = 1, 2, \cdots, N$, a union of any k sets in Ω, $\overset{k}{\cup} S_i$, contains at least k distinct elements, i.e., $|\overset{k}{\cup} S_i| \geq k$, where $\mid S \mid$ represents the number of distinct elements in S.*

Let I_i be a set, where elements in I_i represent the distinct destinations of the cells arrived at input i in one frame. Thus, a perfect matching is in fact an SDR of the family of sets $I = \{I_1, I_2, \cdots, I_N\}$.

Lemma 6 *Let $\overset{k}{\cup} I_i$ be a union of any k sets in I. The following relation holds for any traffic pattern which satisfies Assumption 1:*

$$|\overset{k}{\cup} I_i| \geq k,$$

i.e., the number of distinct destinations of cells belonging to one frame in k input buffers is at least k.

Proof: This is proved by contradiction. Assume that the statement is not true, i.e., the number of distinct destinations of cells belonging to one frame in any k input buffers could be less than k. Since the total number of cells belonging to the frame in these k input buffers is $k \cdot M$, at least one output link must be receiving more than M cells, thus violating Assumption 1. Therefore, the Lemma must be true.

Let $O = \{O_1, O_2, \cdots, O_N\}$ be a family of sets, where O_i is a set, whose elements are the distinct inputs from which cells belonging to one frame are directed to output i.

Lemma 7 *Let $\overset{k}{\cup} O_i$ be a union of any k sets in O. The following relation holds for any traffic pattern which satisfies Assumption 1:*

$$|\overset{k}{\cup} O_i| \geq k,$$

i.e., the number of distinct sources (inputs) of the cells received by k outputs in one frame is at least k.

The proof is the similar to that of Lemma 6.

Lemma 8 *Let Γ be a set of any k inputs, and Θ be a set of any k outputs, where $k = 1, 2, \cdots, N$. If cells belonging to one frame in the k input buffers in Γ are directed to the k outputs in Θ exclusively, the k outputs in Θ can only receive cells from these k inputs in Γ during the frame given that the traffic pattern satisfies Assumption 1, and vice versa.*

Proof: This is proved by contradiction. Assume that the statement is not true, i.e., under the condition that cells belonging to one frame in k input buffers in Γ are directed to the k outputs in Θ exclusively, the k outputs in Θ can still receive cells from inputs which are not in Γ. According to Assumption 1, the number of cells which can be transmitted to any k outputs or from any k inputs in each frame is exactly $k \cdot M$. If the k outputs in Θ receive cells from inputs which are not in Γ, the number of cells transmitted to these k outputs from the k inputs in Γ must be less than $k \cdot M$. Thus, some of the cells in the k inputs in Γ must be transmitted to some outputs not in Θ, which contradicts our condition, and therefore, the lemma must be true. Similarly, if cells received by the k outputs in Θ all come from the k inputs in Γ during one frame, the cells stored in these k input buffers in Γ can only be directed to the k outputs in Θ in that frame.

Lemma 9 *A perfect matching always exists if the traffic pattern satisfies Assumption 1*

Proof: This lemma follows directly from Lemma 6. Since $|\overset{k}{\cup} I_i \;| \geq k$ if the traffic pattern satisfies Assumption 1, according to Philip Hall's Theorem, an SDR exists for the family of sets I, which is in fact a perfect matching.

Theorem 2 *SSF can guarantee strict sense 100% throughput, bounded end-to-end delay, and bounded end-to-end delay jitter.*

Proof: From Lemma 9, a perfect matching exists provided Assumption 1 is satisfied, i.e.,

$$\sum_i C_{i,j} = M \quad \mathit{and} \quad \sum_j C_{i,j} = M \quad \forall i, j = 1, 2, \cdots, N,$$

where $C_{i,j}$ is the number of cells which arrive at input i and directed to output j in one frame.

Once a perfect matching is found in a time slot, the rest of the cells belonging to that frame still satisfy Assumption 1, except that instead of M there are $M - 1$ cells left in each input buffer. Therefore, a perfect matching can be found in each

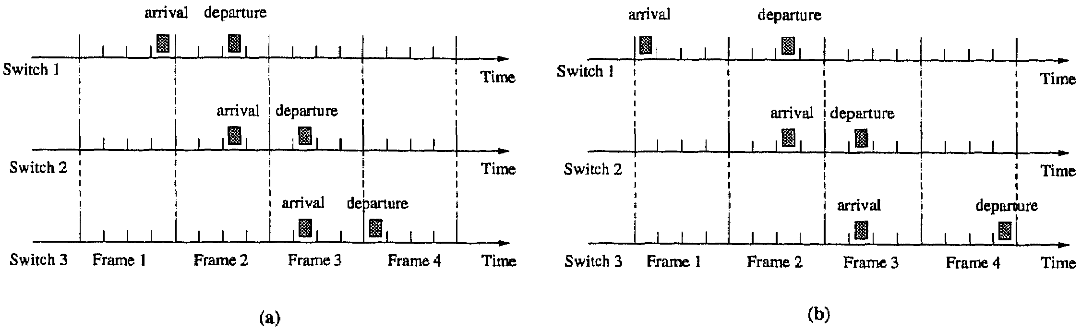


Figure 4.7 Two extreme cases of cell transmission: (a) the cell experiences an end-to-end delay of $(n-1)T+\tau$. (b) the cell experiences an end-to-end delay of $(n+1)T-\tau$.

time slot until all the cells belonging to that frame are scheduled, and thus 100% throughput is achieved.

Noting that cells arrived during one frame can be transmitted in the next frame, the following are two extreme cases of cell transmission for a connection consisting of n concatenated switches: a cell arrived in the last time slot of frame f at the first switch is transmitted in the first time slot of frame $f+n$ at the nth switch resulting in an end-to-end delay of $(n-1)T+\tau$, where τ is the time duration of one time slot, and a cell arrived in the first time slot of frame f at the first switch is transmitted in the last time slot of frame $f+n$ at the nth switch resulting in an end-to-end delay of $(n+1)T-\tau$, as shown in Figure 4.7. Thus, the end-to-end delay and end-to-end delay jitter of a connection are bounded by $(n+1)T-\tau$ and $2(T-\tau)$, respectively.

4.2.2 Complexity Issues

In this section, we derive the upper bound of the number of effective moves needed to obtain a perfect matching.

Consider the following $N \times N$ traffic matrix satisfying Assumption 1,

$$W_0 = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

where block A is a $P \times P$ block, D is an $(N-P) \times (N-P)$ block, and all diagonal elements of A are nonzero.

The rows and columns in W_0 are associated with the inputs and outputs, respectively. Assume that row i is corresponding to input k, and column j is corresponding to output l, where $1 \leq i, j, k, l \leq N$. Therefore, a nonzero element $w_{i,j}$ indicates that input k has cells directed to output l.

Lemma 10 *If D is an $(N-P) \times (N-P)$ "zero" block, the number of nonzero columns in block C is at least $N-P+1$, and similarly, the number of nonzero rows in block B is at least $N-P+1$.*

Proof: There are $N-P$ rows in blocks C and D. According to Lemma 6, cells stored in the inputs corresponding to these rows are directed to at least $N-P$ distinct outputs, i.e., there are at least $N-P$ nonzero columns in blocks C and D. Since D is a "zero" block, there are at least $N-P$ nonzero columns in block C. However, if the number of nonzero columns in block C is $N-P$, i.e., cells stored in the inputs corresponding to the $N-P$ rows are directed to $N-P$ distinct outputs, according to Lemma 8, the $N-P$ outputs corresponding to the $N-P$ nonzero columns in block C can only receive cells from these $N-P$ inputs. Hence, columns in block A corresponding to these $N-P$ outputs must be zero, thus conflicting with the condition that the diagonal elements of A are all nonzero. Therefore, the number of nonzero columns in block C is at least $N-P+1$. Similarly, the number of nonzero columns in block B is at least $N-P+1$.

In the previous section, we claimed that block F in the traffic matrix W_1 must have some nonzero elements. The proof is given by the following Lemma. Note that in the scenario of a multiple effective move, the traffic matrix is decomposed recursively, and a residue matrix is constructed in each iteration of the recursive procedure until a nonzero element $w_{j,k}$ corresponding to a nonzero element $w_{i,j}$ in the residue matrix is found in block G, as shown in Figure 4.8.

Lemma 11 *Consider a traffic matrix which satisfies Assumption 1. Any residue matrix F associated with an $(N-P) \times (N-P)$ all-zero block D must have at least $N-P+1$ nonzero columns.*

Proof: Consider the residue matrix constructed in the first iteration of the recursive procedure. For convenience, the traffic matrix W_1 is written again below:

$$W_1 = \begin{bmatrix} A_1 & X & G \\ F & A_2 & 0 \\ 0 & E & D \end{bmatrix},$$

where D is an $(N-P) \times (N-P)$ "zero" block, and diagonal elements of A_1 and A_2 are all nonzero. By virtue of the procedure in obtaining a matched matrix, the block below a residue matrix is always a "zero" block, and the extended rows of block F constitute a zero block in the area above D.

First, we prove that there are at least $N-P+1$ **columns** in block F. According to Lemma 10, there are at least $N-P+1$ nonzero columns in block C and $N-P+1$ nonzero rows in block B. Thus, if the number of zero columns in block C is less than $N-P+1$, there must exist at least one nonzero element $w_{j,k}$ in block B corresponding to a nonzero element $w_{i,j}$ in block C, which results in a double effective move, and no residue matrix is needed to be constructed. Thus, residue matrix F is only necessary when the number of zero columns in C is larger than or equal to $N-P+1$, i.e., there are at least $N-P+1$ **columns** in block F.

Let m be the number of columns in F, i.e., block A_1 is an $m \times m$ block, and n be the number of nonzero columns in F, and thus there are $m-n$ zero columns in F. Suppose cells received by the outputs corresponding to these $m-n$ extended columns of F and the $N-P$ extended columns of G come from d distinct inputs.

Since all nonzero elements in these extended columns are in blocks A_1 and G, the number of distinct inputs d is no more than m, i.e., $d \leq m$.

Consider $n < N - P$: By Lemma 7, since $m - n + N - P > m$ for $n < N - P$, the number of distinct inputs d corresponding to these $m - n + N - P$ outputs must be greater than or equal to $m - n + N - P$, i.e., greater than m. This contradicts the above, and thus $n \geq N - P$.

Consider $n = N - P$: By Lemma 7, since $m - n + N - P = m$ for $n = N - P$, the number of distinct inputs d corresponding to these $m - n + N - P$ outputs must be greater than or equal to $m - n + N - P$, i.e., $d \geq m$. On the other hand, d must be less than or equal to m as shown above, which implies that d must be equal to m. However, by Lemma 8, the cells stored in these $d = m$ inputs corresponding to the rows in A_1 are exclusively directed to the m outputs corresponding to the $m - n$ columns in block A_1 and the $N - P$ columns in block G, thus resulting in n zero columns in block A_1. This conflicts with the condition that all diagonal elements in A_1 are nonzero. Therefore, n is at least $N - P + 1$.

Noting that the elements below any residue matrix and the elements below block G are always zero, the proof of the lemma for a residue matrix constructed in any other iteration of the recursive procedure is similar to that for the residue matrix constructed in the first iteration, and thus the lemma is proved.

Theorem 3 *Let k, where $k \geq 1$, be the number of effective moves within which the size of block A is guaranteed to increase from $P \times P$ to $(P+1) \times (P+1)$ in SSF, where $P = 0, 1, 2, \cdots, N-1$. The following relation between k and P holds for any traffic patterns satisfying Assumption 1.*

$$\frac{(k-1) \cdot (N+1)}{k} \leq P < \frac{k \cdot (N+1)}{k+1}.$$

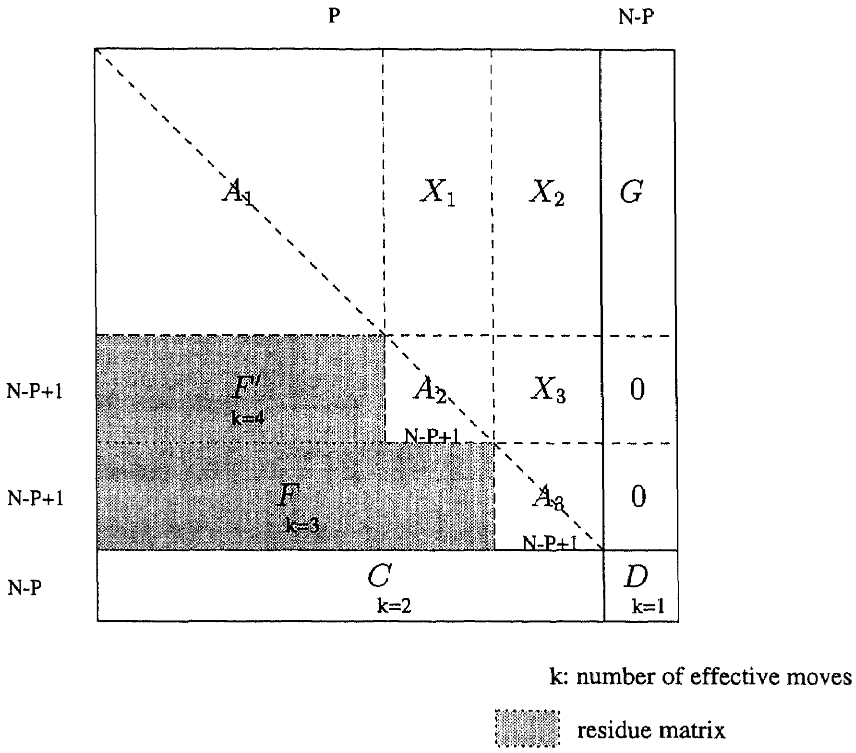


Figure 4.8 Decomposition of a traffic matrix

Proof: According to Lemma 11, the number of nonzero columns in a residue matrix is at least $N - P + 1$ if the traffic pattern satisfies Assumption 1. The larger the number, the smaller number of decompositions of the traffic matrix is needed to guarantee that a nonzero element $w_{j,h}$ in G corresponding to a nonzero element $w_{i,j}$ in the residue matrix can be found, i.e., a smaller number of effective moves are needed. To derive the number of effective moves within which the size of block A is guaranteed to increase by one, the worst case is considered. Thus, we assume that there are $N - P + 1$ nonzero columns in each residue matrix as shown in Figure 4.8.

Note that the size of a residue matrix is $(N-P+1) \times (P-(k-2)(N-P+1))$, where k is the number of effective moves required to move a nonzero element to the upper-left corner of block D if a nonzero element $w_{j,h}$ which is corresponding to a nonzero element $w_{i,j}$ in the residue matrix can be found in G, as shown in

Figure 4.8. Since there are $N - P + 1$ nonzero rows in G and $N - P + 1$ nonzero columns in a residue matrix, if the number of columns in the residue matrix is less than $2(N - P + 1)$, i.e., $P - (k - 2)(N - P + 1) < 2(N - P + 1)$, or equivalently, $P < \frac{k \cdot (N+1)}{k+1}$, at least one nonzero element $w_{j,h}$ corresponding to a nonzero element $w_{i,j}$ in that residue matrix can be found in G. Thus, k effective moves are sufficient to increase the size of block A by one if $P < \frac{k \cdot (N+1)}{k+1}$. Similarly, $k - 1$ effective moves are sufficient to increase the size of block A by one if $P < \frac{(k-1) \cdot (N+1)}{k}$. Thus, k effective moves are necessary only if $\frac{(k-1) \cdot (N+1)}{k} \leq P$. Therefore, k effective moves are necessary and sufficient to guarantee to increase the size of block A by one if $\frac{(k-1) \cdot (N+1)}{k} \leq P < \frac{k \cdot (N+1)}{k+1}$.

Theorem 4 *The number of effective moves required to obtain a perfect matching in SSF is bounded by $N(\ln N + O(1))$.*

Proof: From Theorem 3, the size of block A is guaranteed to increase by one within k effective moves if the number of rows (columns) in block A satisfies the following condition,

$$\frac{(k-1) \cdot (N+1)}{k} \leq P < \frac{k \cdot (N+1)}{k+1} \tag{4.1}$$

Equation (4.1) can be rewritten as follows:

$$\frac{P}{N - P + 1} \leq k \leq \frac{N+1}{N - P + 1}$$

Thus, the total number of effective moves $f(N)$ required to obtain a perfect matching in the worst case is $\sum_{P=0}^{N-1} k$ which is bounded by

$$f(N) \;=\; \sum_{P=0}^{N-1} k \leq \sum_{P=0}^{N-1} \frac{N+1}{N - P + 1} \tag{4.2}$$

Rewrite the right part of Equation (4.2) in terms of n, where $n = N - P + 1$.

$$f(N) \;\leq\; (N+1) \sum_{n=2}^{N+1} \frac{1}{n} \leq N \sum_{n=1}^{N} \frac{1}{n} \tag{4.3}$$

Note that the sum of the Harmonic series is given by:

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$$

Thus,

$$f(N) \quad \leq \quad N(\ln N + O(1)) \tag{4.4}$$

i.e., the number of effective moves required to obtain a perfect matching is bounded by $N(\ln N + O(1))$.

CHAPTER 5

CONTRIBUTIONS ON ATM MULTICASTING

Multiway communication involves transferring of data simultaneously from multiple senders to one or more receivers, using a single, shared multicast tree. Such a mechanism can be managed simply by maintaining a separate multicast tree, rooted at each sender. But, this simple scheme does not efficiently utilize the network resources like bandwidth. Moreover, connection management becomes difficult when participants join or leave the connection during the multiway session.

Multiway communication can be supported more efficiently if all the senders share a single multicast tree. Such connections are also called as multipoint-to-multipoint connections. A multicast group can be supported using a single multipoint-to-multipoint connection, even when there are multiple senders. Several multimedia applications like video conference, interactive video games and distributed interactive simulations require this support from the underlying network layers. In a multipoint-to-multipoint connection, data from multiple senders are merged into a single connection at appropriate "merge points" and forwarded towards the receivers.

The routing and signaling protocols specified in the current standards for ATM networks do not support multipoint-to-multipoint connections. Only a rudimentary support for point-to-multipoint connections is specified in the standards. But, recently, several protocols to establish multipoint-to-multipoint connections in ATM networks have been proposed for possible standardization [44, 45, 46]. In this chapter, we assume that one such mechanism is already implemented to establish multipoint-to-multipoint connections.

Based on the above assumption, a single connection identifier (VPI/VCI) is associated with each multipoint-to-multipoint ATM connection. Since VPI/VCI values have a local significance on a given link, a direct implication of this association is that all the ATM cells of this connection, even those from different senders, use the same VPI/VCI value on that particular link. This conserves the VPI/VCI space and the switch resources, while simplifying the signaling mechanisms when there are several simultaneously active multicast groups. This results in a scalable mechanism for supporting multipoint-to-multipoint ATM connections.

At the sender, the ATM cells are generated by the fragmentation of higher layer packets. Each ATM cell, therefore, does not carry information about the sender and the receiver. The ATM Adaptation Layer (AAL) at the receiver is responsible for re-assembling the original higher layer packets from the individual ATM cells. Cells of different packets originating from different senders intended for the same multicast group may get interleaved with each other. Since all the cells use the same VPI/VCI value, the receiver may not be able to uniquely identify the sender of a particular ATM cell. Therefore, the original packet cannot be re-assembled at the receiver. This is called the sender identification problem, associated with multipoint-to-multipoint connections in ATM networks.

Several solutions have been proposed to solve the sender identification problem for multipoint-to-multipoint connections in ATM networks. In this chapter, we provide a systematic study of these solutions. The solutions are classified into two, based on their inability or ability to support interleaving of ATM cells belonging to different packets intended for the same multipoint-to-multipoint connection. We compare the fundamental characteristics of each of these classes of solutions. The factors used for comparison include the buffer requirements, the extra overheads carried within each cell or packet, the complexity of the mechanism, the changes required to existing network components and inter-operability.

VC-Merge and VP-Merge are representative solutions for the two categories of solutions. VC-Merge is fast and scalable, but it requires the use of additional buffers at intermediate merge points. VP-Merge, on the other hand, needs no additional buffers, but its scalability is restricted due to the excessive use of VPI/VCI space. It is therefore desirable to design a scheme that combines the advantages of the VC-Merge and VP-Merge mechanisms. Such a scheme would require very little additional buffers and at the same time, will not be restricted by its use of VPI/VCI space. Design of such schemes is the focus of this chapter. This chapter proposes a generic scheme for merging data from multiple senders onto one or more outgoing links.

The chapter is organized as follows. In Section 1, we categorize and analyze the various solutions to the sender identification problem. Section 2 compares the VC-Merge and VP-Merge schemes under various scenarios. In Section 3, we propose a generic VC-Merge scheme and analyze three mechanisms based on this generic scheme. Section 4 concludes the chapter with the results and the direction for future work.

5.1 Solutions to the Sender Identification Problem

The sender identification problem arises because the receiver may not be able to uniquely identify the source of an ATM cell when the cells from different packets intended for the same multipoint-to-multipoint connection are interleaved. One way of solving the problem is by preventing the interleaving of cells. Since cells from different senders may arrive in any order at an intermediate switch, special mechanisms are required to prevent interleaving of cells. In the next section, we briefly describe some of these mechanisms.

ATM networks, which supports statistically multiplexing, derives some of its advantages due to interleaving of cells. Therefore, it may not be desirable to prevent cell interleaving. In order to support cell interleaving, the identity of the sender of each cell has to be conveyed to the receiver. In a subsequent section, we discuss some of the mechanisms used to convey this information from the sender to the receiver.

5.1.1 Mechanisms that Prevent Cell Interleaving

The mechanisms discussed in this section prevent cell interleaving by sending all the cells of a packet contiguously. The sender can be identified from the reassembled packet at the receiver(s). Note that interleaving of cells belonging to different connections is not restricted by any of these mechanisms.

In the Multicast Server (MCS) approach [47], a centralized multicast server ensures contiguity of all the cells of a packet. The senders of a multipoint-to-multipoint connection first send the data to a pre-assigned multicast server responsible for forwarding the data packets to all the receivers. The scheduler at the MCS prevents interleaving of cells belonging to different packets. This approach can be easily deployed on existing networks supporting point-to-multipoint connections. But, the lack of scalability and single point of congestion and failure are its main disadvantages.

A token-based approach [48] requires a user to possess a token for sending packets to a multipoint-to-multipoint connection, thereby, restricting multiple users from simultaneously sending packets to the same connection. The token is passed on from one sender to another. Though this scheme works especially well for links with limited bandwidth, it does not scale well to large number of senders because of the overheads involved in token-passing and recovery of lost tokens.

Buffering of cells at appropriate "merge points" and intelligent scheduling can be used to prevent cell interleaving. In one possible implementation called the store and forward VC-Merge mechanism [49], an intermediate switch buffers all the cells of a packet till the entire packet reaches the switch. The cells are then scheduled to be contiguously forwarded towards the destination(s) using the entire bandwidth allocated for this connection. Note that cells of a packet can interleave with cells of packets intended for a different connection, which distinguishes this scheme from traditional packet switching. An alternate implementation called the virtual cut-through VC-Merge scheme allows an intermediate switch to schedule partial packets for forwarding. But once a packet is scheduled, cells of other packets intended for the same multipoint-to-multipoint connection have to be buffered till the scheduled packet is completely forwarded. This wait depends on the rate at which the scheduled packet is arriving.

The VC-Merge approaches described here are efficient because the cells need not be reassembled at each intermediate switch. Instead, the end-of-packet (EOP) indicator as specified in AAL5 is used to detect the end of a particular packet. VC-Merge has very little computational overhead, but needs additional buffers at appropriate intermediate switches, whose size depends on the number of senders, the traffic characteristics and the packet sizes. In subsequent sections, we study the buffer requirements and propose mechanisms to reduce the amount of additional buffers.

5.1.2 Mechanisms that Support ATM Cell Interleaving

In order to support ATM cell interleaving for multipoint-to-multipoint connections, the identity of the sender must be included in each cell. In the VC-Mesh approach [47], each sender of a multipoint-to-multipoint connection establishes a separate point-to-multipoint connection identified by distinct VPI/VCI field. This allows cell interleaving, but complicates dynamic changes to the set of senders and receivers due to the maintenance of large number of connection states.

Alternately, the sender information can be encoded in the 10-bit multiplex ID (MID) field of AAL3/4 ATM Adaptation Layer. Each sender has to be assigned a unique MID value using some additional mechanisms, thereby, restricting the number of senders to 1024. AAL3/4 is not widely used because the payload is limited to only 44 bytes (as defined in AAL3/4), limiting the effective utilization to 83%.

In the VP-Merge technique, only the VPI field is used to identify a multipoint-to-multipoint connection and a VCI value, unique to each sender within the VPI value, is used to identify the sender. The cells are switched on the VPI value and the VCI value is carried undisturbed. This scheme can be implemented on existing VP switches, but the scalability is limited because the number of independent multipoint-to-multipoint connections on a given link can be at most 4096 ($= 2^{12}$).

In the widely prevalent AAL5 adaptation layer standard, there is no field for sender information in the header or the payload. A new AAL (currently non-standard) that includes the sender information as part of the 48 byte payload can be proposed for multipoint-to-multipoint connections. In one possible implementation, the value of first two bytes of the ATM payload, uniquely assigned to each sender, can be used to identify the sender. In this implementation, the actual payload is only 46 bytes long, thereby, limiting the effective utilization to 86.79%. Proposing this new protocol may lead to incompatibility with existing infrastructure. Further, a single bit error in the non error-corrected sender value will affect packets from two different senders.

A scheme proposed in [50] facilitates the use of standard AAL5 protocols by introducing a Resource Management (RM) cell to carry sender identities. Each RM cell contains identities of the senders of the following few ATM cells intended for a particular multipoint-to-multipoint connection. The receiver interprets the RM cell to correctly identify the senders of these ATM cells. The RM cell on a given link is significant only to the switches at either end of the link. The RM cell is therefore created at each switch in accordance with the scheduling policy of that switch. This prevents the propagation of wrong information due to lost cells. Though the effective utilization for two-byte long sender identities is 86.79%, the performance may be affected due to the creation of RM cell at each switch. Since the loss of an RM cell can affect several packets, the RM cells are sent with CLP-bit 0 to minimize the chance of losing these cells.

5.2 Comparison between VC-Merge and VP-Merge

We now compare the two categories of solutions described in the preceding sections. The VC-Merge techniques and the VP-Merge techniques are used as representative techniques for the two categories of solutions for solving the sender identification problem. The VC-Merge uses a coarser granularity of multiplexing based on packet interleaving, while VP-Merge uses a finer granularity of multiplexing based on cell interleaving.

VC-Merge has been accepted to support multipoint-to-point connections in future versions of PNNI specifications. We believe that VP-Merge is an equally good alternative. Connections based on VP-Merge technique closely resemble ATM connections because it supports statistical multiplexing of ATM cells. Moreover, the buffer requirement for VP-Merge is much smaller than that for VC-Merge. To study the buffer requirements, we simulated a typical scenario for VP-Merge and VC-Merge at a merge point. A merge point is a switch that has at least two incoming links for the multiway connection and one outgoing link that is different from the two incoming links.

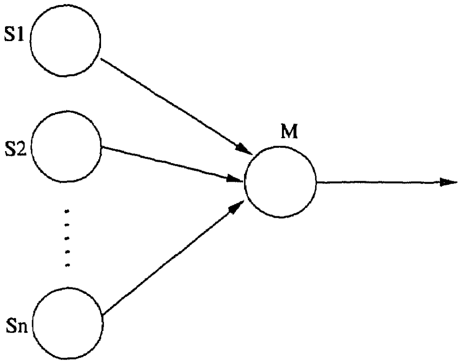
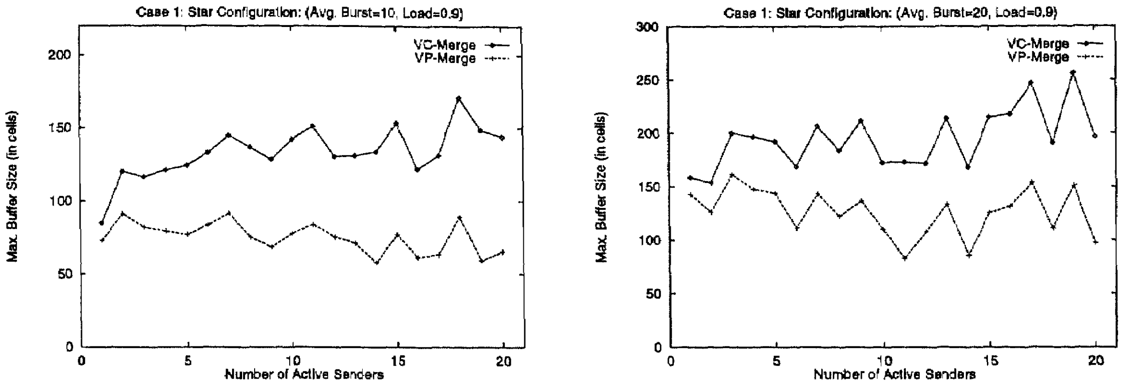


Figure 5.1 Star configuration

5.2.1 Simulation Study

We studied the buffer requirements at a merge point for multipoint-to-point connections. We first considered a merge point with several senders, as shown in Figure 5.1. We call this the Star configuration. There are several senders S1, S2 . . . Sn, sending data towards a merge point M. The cells from these senders are merged and sent out from the merge point. We analyzed the amount of buffer required at the merge point by varying the number of senders. Each sender is assumed to be a bursty source. All the senders are assumed to be identical sources and the capacity of the outgoing link from the merge point is normalized to 1. Therefore, for a stable system, it is required that the sum of the loads of the senders is less than 1. We assume that there are no cells lost in transit.

A bursty source, characterized by the peak cell rate (P), the average cell rate (A) and the average number of cells per burst (B), can be modeled as an ON-OFF source in the discrete-time domain (two-state Markov Modulated Deterministic Process (MMDP) [38]), as shown in Figure 3.2. 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 (P). In a discrete-time domain, the source can independently shift from one state to the other only at the end of a time-slot.



(a) Burst Length = 10

(b) Burst Length = 20

Figure 5.2 Star Configuration: Case 1: VC-Merge vs VP-Merge

Using this model for the sources, we simulated the buffer requirements at the merge point. We simulated three cases, which are described here.

- Case 1: The entire packet is transmitted at the end of a burst. Further, the incoming links are slow links and the outgoing link from the merge point is a faster link.

- Case 2: The entire packet is transmitted at the end of a burst. But, the incoming and outgoing links have the same speed.

- Case 3: The packets are of fixed length and the complete packet may not be transmitted at the end of a burst. The incoming and outgoing links have the same speed.

The plot of comparison for the first case is shown in Figure 5.2(a). In this figure, the average burst length is 10. From the figure, it is clear that, on an average, VC-Merge technique uses 83% more buffer than the VP-merge technique. We repeated the same experiment using different values of average burst length. As the average

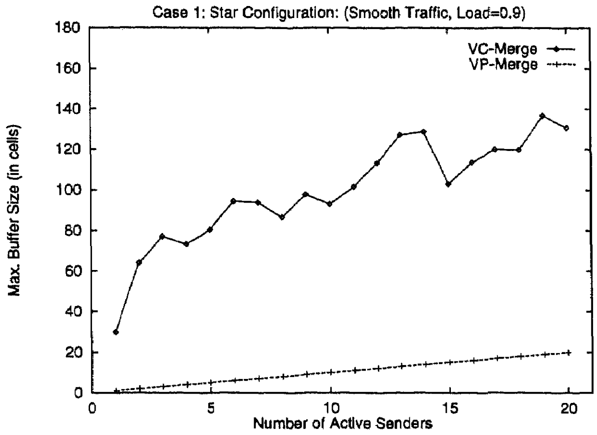


Figure 5.3 Star Configuration: Case 1: VC-Merge vs VP-Merge: Smooth

burst length increases, the buffer requirement for the VP-Merge technique increases correspondingly. This leads to a decrease in the overall percentage increase in buffer for the VC-Merge technique. The plot for average burst length = 20 is given in Figure 5.2(b). From the figure, we can see that the average increase in buffer requirement is about 59% for VC-Merge technique over the VP-Merge technique. The difference becomes even more pronounced as the traffic becomes smoother as shown in Figure 5.3. This is due to the fact that very little buffering is required for the VP-Merge technique when the traffic is smooth.

In the second case, we studied the buffer requirements when the incoming and outgoing links have the same speed. In this scenario, the peak cell rate of the sender is equal to 1, the normalized outgoing link rate. The average cell rate (the load) is adjusted as the number of senders increase to ensure the stability of the system. Again, it is assumed that the entire packet is generated during a burst. The buffer requirements when the average burst length is 10 are plotted in Figure 5.4.

In this scenario, the VC-Merge requires about 27% more buffer on the average. The difference is smaller than the previous case because entire packets reach the merge point faster because of the faster incoming links. Therefore, cells on an incoming link become available for switching at a faster rate. The difference decreases further as the average burst length increases due to the same reasons as explained in the previous case.

Next, in the third case, we studied the buffer requirements when the entire packet is not generated during a burst. The packets are assumed to be of fixed length and consisting of 30 data cells. Again, the incoming and the outgoing links have the same speed. The results of the plot for average burst length = 10 are shown in Figure 5.5. In this case, the buffer requirement for VC-Merge increases because the entire packet may not arrive in a single burst. The buffer requirement for VP-Merge remains the same as in the previous case.

5.2.2 Summary of the Results

From the simulation studies, we can conclude that VP-Merge has a clear advantage over VC-Merge based on the buffer requirements. The advantages are more pronounced for smoother traffic than for bursty traffic. In this respect, we concur with the opinions expressed in [51]. So, VP-Merge is a better alternative to VC-Merge with respect to buffer requirements. But, the main problem with VP-Merge is its poor scalability. Moreover, it is difficult to implement congestion control mechanisms like Early Packet Discard (EPD) in a VP-Merged connection.

5.3 Improved VC-Merge Mechanisms

The ATM Forum has accepted VC-Merge to support multipoint-to-point connections in future versions of PNNI specifications. In this section, we describe some improvements to the traditional VC-Merge schemes that reduce the buffer requirements at the intermediate switches at the cost of increased utilization of the VPI/VCI space. Since the number of different connections supported on a given link is much less than

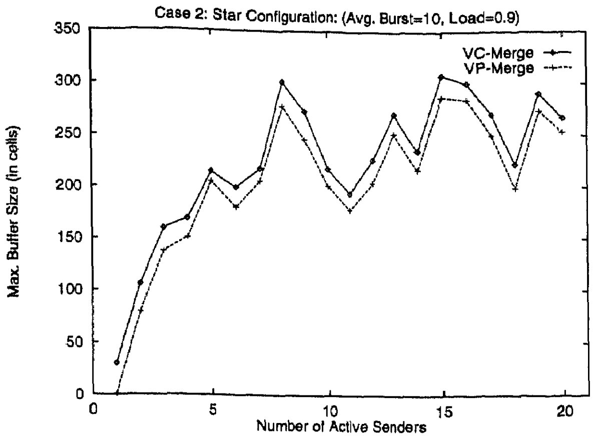


Figure 5.4 Star Configuration: Case 2: VC-Merge vs VP-Merge

the available VPI/VCI space, this increased utilization does not affect the scalability of the proposed mechanisms.

5.3.1 Multiple VC-merge Mechanisms

We propose some improvements to the VC-Merge approach to minimize the the buffer requirements at the intermediate switches. These improvements, referred as multiple VC-Merge mechanism, adopt the use of multiple VPI/VCI values for a particular multipoint-to-multipoint connection. In some sense, each connection has multiple connection identifiers. Note that the VPI/VCI values still retain local significance on a given link. Some signaling protocol is required to map multiple connection identifiers to the same multipoint-to-multipoint connection. This could be done either a priori during the connection set up phase or dynamically depending on the performance of the system. The details of the signaling mechanism is outside the scope of this chapter.

The multiple VC-Merge mechanism does not affect systems that use only store and forward VC-Merge scheme. But, it has a great impact on systems that incorporate virtual cut-through VC-Merge mechanisms. On these systems, though this

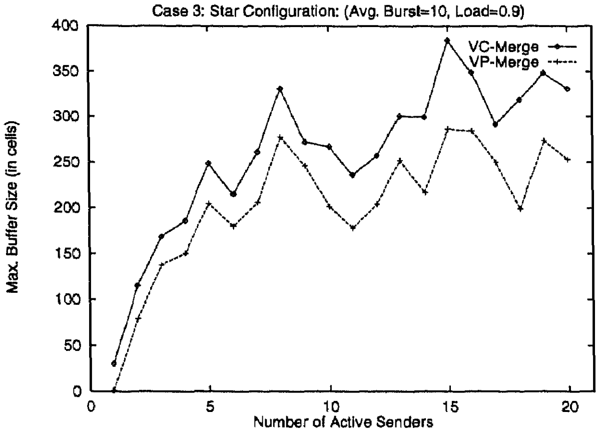


Figure 5.5 Star Configuration: Case 3: VC-Merge vs VP-Merge

scheme increases the VPI/VCI space for each multipoint-to-multipoint connection, it improves the throughput and reduces the buffer requirements at intermediate switches. This improvement is mainly due to the fact that the multiple VC-Merge mechanism restricts interleaving of cells belonging to different packets only on the same connection identifier, but permits interleaving of cells on different connection identifiers referring to the same multipoint-to-multipoint connection.

The multiple VC-Merge mechanism is a generalized merge mechanism. On one extreme, if there is exactly one connection identifier for a particular multipoint-to-multipoint connection, the buffer requirements and the characteristics of the connection resemble the traditional VC-Merge mechanism. On the other extreme, when the number of connection identifiers equals the number of senders for a multipoint-to-multipoint connection, the buffer requirements and the characteristics of the connection resemble the traditional VP-Merge mechanism. Typically, multiple VC-Merge mechanism operates between these two extremes.

We now discuss two possible implementation schemes for the multiple VC-Merge mechanism in a system that supports only virtual cut-through mechanism.

5.3.1.1 Fixed Multiple VC-Merge Mechanism (FMVC): In this implementation scheme, a switch that acts as a merge point for a particular multipoint-to-multipoint connection statically assigns one of the corresponding connection identifiers to each sender. All the cells originating from a sender intended for that connection are forwarded on the outgoing link(s) using the assigned connection identifier. The identifier is assigned when the sender joins the connection and remains fixed till the sender leaves the connection. If the number of connection identifiers is less than the number of senders, more than one sender will be assigned the same connection identifier. This results in partitioning the set of senders into identifier groups, where each identifier group of senders is assigned a particular connection identifier. Though it is possible to interleave cells belonging to packets that originate from senders that are in different identifier groups, it is not possible to interleave cells belonging to packets originating from two senders that are in the same identifier group. Since the partitioning and assignments are fixed, it may happen that some cells belonging to particular identifier group have to be buffered even though there are no active cells belonging to some other identifier group.

5.3.1.2 Dynamic Multiple VC-Merge Mechanism (DMVC): In the dynamic implementation, a switch that acts as a merge point for a particular multipoint-to-multipoint connection maintains the set of unassigned connection identifiers on the outgoing link(s) pertaining to that connection. When the first cell of a packet intended for that connection arrives at this merge point, one of the unassigned connection identifiers from that set is assigned to this packet. This identifier is then removed from the set. All the cells of this packet use this assigned identifier on the outgoing link(s). Once the entire packet of cells is transmitted, the assigned identifier is released back to the set.

If there are no unassigned connection identifiers when a packet arrives at a merge point, the cells of that packet are buffered till one of the identifiers becomes free. This results in the efficient utilization of the connection identifiers.

It is possible to implement DMVC in a network only if all the switches are capable of mapping multiple connection identifiers to the same logical queue. Typically, at a given switch, all the cells arriving on a particular input port having the same connection identifier are assigned to the same logical queue (either input or output). In a multiple VC-Merge scenario, the switch must be capable of mapping multiple connection identifiers to the same logical queue. If all the switches in a network do not have the capability of mapping multiple connection identifiers to the same logical queue, then it is possible that packets originating from a particular sender may arrive out of sequence at a receiver. Though it is possible to resequence the packets using higher layer protocols, this is against the philosophy of connection-oriented networks like ATM. In networks comprising of some switches that cannot map multiple connection identifiers to the same logical queue, a mechanism like FMVC has to be implemented to solve the sequencing problem.

5.3.1.3 Selective Multiple VC-Merge Mechanism (SMVC): The FMVC and DMVC implementations described in the previous sections do not impact the store and forward VC-Merge mechanism. These schemes can be enhanced by maintaining an additional connection identifier for store and forward VC-Merge mechanism. This scheme is called the Selective Multiple VC-Merge mechanism (SMVC). In the simplest implementation of SMVC, two connection identifiers are maintained for each multipoint-to-multipoint connection. One connection identifier is used for virtual cut-through VC-Merge scheme and the other is used for store and forward VC-Merge scheme. When all the cells of a packet intended for a

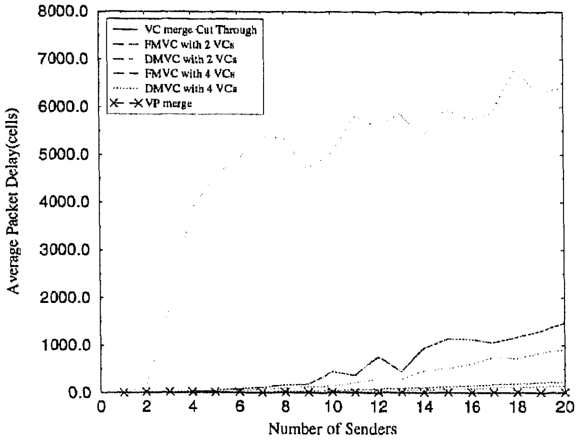


Figure 5.6 Comparison of DMVC and FMVC with virtual cut-through VC-Merge

multipoint-to-multipoint connection is available at a merge point, the store and forward connection identifier is used to forward these cells on the outgoing link(s). It is not necessary to initiate forwarding using the virtual cut-through mechanism because the bandwidth allocated for this multipoint-to-multipoint connection is fully utilized for the store and forward mechanism. If only partial packets are available, one of the partial packets is scheduled on the outgoing link(s) using the virtual cut-through connection identifier. All the cells of this packet will be eventually forwarded using this connection identifier. This improves the link utilization and reduces the buffer requirements at the merge point.

5.3.2 Simulation Results

We performed extensive simulations to study the buffer requirements for each of the proposed improvements. We focus on the buffer requirements at a particular Merge Point of a single multipoint-to-multipoint connection. Specifically, we studied the "star" configuration as shown in Figure 5.1 of Section 5.2.1. In order to study the effect of heterogeneous senders on the buffer requirements, we used two sets of senders, the fast and the slow senders. The fast senders generate cells at twice the

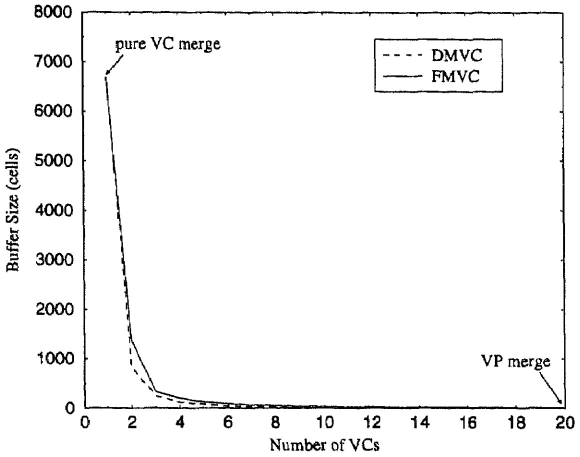


Figure 5.7 Comparison of DMVC and FMVC with increasing number of identifiers

rate as the slow senders. In order to prevent buffer overflow, we maintained the total load on the outgoing link at the merge point at 90% of its capacity. For lack of space, we present the results from only one scenario used in our simulations. In this scenario, we assumed that the average burst length of each source is 10 cells and the burstiness factor is 2.

In Figure 5.6, we compare the DMVC and FMVC mechanisms with respect to the virtual cut-through VC-Merge technique. Since the sources are not very bursty(burstiness = 2) and the load on the outgoing link is only 90% utilized, VP-Merge mechanism requires very little buffers at the merge point. In this scenario, the buffer requirements for virtual cut-through mechanism is very large, as is evident in Figure 5.6. This is due to the fact that when cells from a slow source are scheduled on the outgoing link, cells from other sources have to be buffered till an entire packet of cells is transmitted out. The slow source does not efficiently utilize the outgoing link. The multiple VC-Merge techniques, DMVC and FMVC, reduce the buffer requirements by about 80% just by using two connection identifiers per connection. This reduction results from the ability to schedule two packets, one using each connection identifier, simultaneously. The utilization of the outgoing link improves

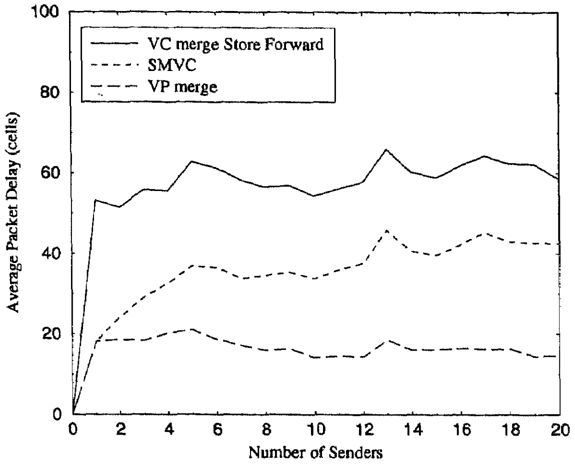


Figure 5.8 Comparison of SMVC with store and forward VC-Merge

by a great extent. As expected, the DMVC technique does a little better than the FMVC. The buffer improvements are marginal when the number of identifiers is increased to 4.

We now compare the buffer requirements using DMVC and FMVC techniques as the number of connection identifiers increases. The plot of comparison is shown in Figure 5.7. In this plot, the number of senders was fixed at 20. At one extreme of the plot, when there is exactly one connection identifier per connection, the buffer requirements are identical to that of virtual cut-through VC-Merge technique. At the other extreme, when the number of connection identifiers equals the number of senders, the buffer requirements are similar to the VP-Merge technique. In a typical multiple VC-Merge scenario, we operate between the two extremes. From the figure, it is clear that the buffer reduction is phenomenal when the number connection identifiers is increased to 2. There is further improvement as the number increases to 4. But, there is hardly any improvement beyond 10 connection identifiers per connection.

Figure 5.8 compares the average buffer size required by SMVC with those required by VP-Merge and store and forward VC-Merge techniques. From the plot, it is evident that the buffer requirements for SMVC are about 50% less than that for the VC-Merge technique. This is due to the use of two connection identifiers for the same connection. In the extreme case when there is only one active sender, the buffer requirements for VP-Merge and SMVC are identical because both can forward the cells of the packets immediately. On the other hand, the VC-Merge scheme has to buffer the cells until the entire packet has reached the Merge Point. In a general sense, use of two connection identifiers improves the buffer requirements at the Merge Point by about 50%.

5.4 Conclusions

We propose three new schemes for improving the performance of traditional VC-Merge techniques for the support of multipoint-to-multipoint connections in ATM networks. Aptly named DMVC, FMVC and SMVC, these mechanisms define a generic scheme for merging data from multiple senders onto one or more outgoing links. The mechanisms combine the advantages of VP-Merge and VC-Merge in terms of the effective conservation of the VPI/VCI space and the reduction in the buffer requirements at intermediate merge points. Using extensive simulations, we show that there is a 80% reduction in buffer requirements just by using two connection identifiers per connection. These schemes, thus, operate between the two extremes of VC-Merge and VP-Merge. Future work will involve the design of signaling mechanisms for the support of these generic schemes.

CHAPTER 6

SUMMARY AND FUTURE RESEARCH

In this dissertation, we have discussed QoS features of input-queued cell switches and have proposed several algorithms to provide QoS features and guarantees for input-queued cell switches.

In Chapter 2, we modeled and analyzed the back pressure with uniform independent Bernoulli traffic load, and showed that back pressure occurs with high probability under loaded traffic. We also derived the average queue length at the input buffer. To address the above issues in input-queued switches, we proposed a maximum size matching based algorithm, referred to as min-max fair input queueing (MFIQ), to minimize the additional delay caused by back pressure, and at the same time to provide fair service among competing sessions.

Although maximum size matching based algorithms work well for uniform traffic, they are not stable for non-uniform traffic. In Chapter 3, we proposed two maximum weight matching based algorithms which are stable for both uniform and non-uniform traffic, and at the same time provide some QoS features. By setting the weight of an edge in the bipartite graph to the normalized queue length of the corresponding VOQ, the longest normalized queue first (LNQF) provides fairer service than LQF and better traffic shape than OCF. The stability of LNQF is also proven. In earliest delay due date first matching (EDDFM), the weight is set to the delay due date, and thus forces the cells with the earliest delay due date to have the highest priority to receive service. Simulation results showed that EDDDFM has lower probability of delay over due than LQF, LNQF, and OCF.

In Chapter 4, a frame based scheduling algorithm, referred to as Store-Sort-and-Forward (SSF), was proposed to provide QoS guarantees for input-queued switches without requiring speedup. SSF uses a framing strategy in which the time axis is divided into constant-length frames, each made up of an integer multiple of time slots. Cells arrived during a frame are first held in the input buffers, and are then "sorted-and-transmitted" within the next frame. A bandwidth allocation strategy and a cell admission policy are adopted to regulate the traffic to conform to the (r, T) traffic model. A strict sense 100% throughput is proved to be achievable by rearranging the cell transmission orders in each input buffer, and a sorting algorithm was proposed to order the cell transmission. The delay and delay jitter are bounded by the transmission time of one frame. It was proved that a perfect matching can be achieved within $N(\ln N + O(1))$ effective moves.

Noting that the (r, T) traffic model is a strong assumption, and thus incorporating a more realistic traffic model such as leaky bucket in SSF is an interesting problem. Since continuous frames in SSF may have the similar traffic patterns, it is possible to derive a faster algorithm by using configuration information obtained from the previous frame. Our future effort will be focusing on these issues as well as hardware implementation of SSF.

REFERENCES

1. F. Tobagi, "Fast packet switch architectures for braodband integrated services digital networks," *Proceedings of the IEEE*, vol. 78, pp. 133–167, Nov. 1990.

2. A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, Jun. 1993.

3. J. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proceedings of the ACM SIGCOMM*, pp. 143–156, Aug. 1996.

4. S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proceedings of IEEE INFOCOM*, pp. 636–646, 1994.

5. L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching," *IEEE Transactions on Computer Systems*, vol. 9, no. 2, pp. 101–124, Mar. 1991.

6. J. Bennett and H. Zhang, "WF^2Q: Worst-case fair weighted fair queueing," in *Proceedings of IEEE INFOCOM*, pp. 120–128, Mar. 1996.

7. S. J. Golestani, "A stop-and-go queueing framework for congestion management," in *Proceedings of the ACM SIGCOMM*, pp. 8–18, 1990.

8. H. Zhang, "Providing end-to-end performance guarantees using non-work-conserving disciplines," *Computer Communications: Special Issue on System Support for Multimedia Computing*, vol. 18, no. 10, pp. 769–781, Oct., 1995.

9. H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, Oct. 1995.

10. Ascend Communications, *GRF Family of Switches*, www.ascend.com. (17 Oct. 1997)

11. Digital Equipment Corporation, *GIGA switch*, www.networks.digital.com. (17 Oct. 1997)

12. T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *IEEE Transactions on Computer Systems*, vol. 11, no. 4, pp. 319–352, Nov. 1993.

13. N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings of IEEE INFOCOM*, pp. 296–302, 1996.

14. N. McKeown, J. Walrand, and P. Varaiya, "Scheduling cells in an input-queued switch," *IEE Electronics Letters*, pp. 2174–2175, Dec. 9th 1993.

15. A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," in *Proceedings of the ICCCN*, pp. 226–231, Oct. 1996.

16. A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proceedings of IEEE INFOCOM*, pp. 792–799, 1998.

17. S. Li, J. Chen, and N. Ansari, "Min-max fair input queueing with back pressure," in *Proceedings of IEEE ICATM*, pp. 252–259, 1998.

18. S. Li and N. Ansari, "Provisioning QoS features for input-queued switches," *IEE Electronics Letters*, vol. 34, no. 19, pp. 1826–1827, Sept. 1998.

19. S. Li and N. Ansari, "Input queued switching with QoS guarantees," in *Proceedings of IEEE INFOCOM*, pp. 1152–1159, Mar. 1999.

20. S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," Tech. Rep. CSL-TR-98-758, Stanford University, 1998.

21. I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch," in *Proceedings of IWQoS*, pp. 218–224, 1998.

22. M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. COM-35, pp. 1347–1356, Dec. 1987.

23. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw Hill, New York, 1989.

24. D. Stilliadis and A. Verma, "Providing Bandwidth Guarantees in an Input-Buffered Crossbar Switch," in *Proceedings of IEEE INFOCOM*, pp. 960–968, Apr. 1995.

25. N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, " The Tiny Tera: A Packet Switch Core," *IEEE Micro*, pp. 26–33, Jan. 1997.

26. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Internetworking: Research and Experience*, vol. 1, pp. 3–26, 1990.

27. C. Kalmanek, H. Kanakia, and S. Keshav, "Rate controlled servers for very highspeed networks," in *Proceedings of IEEE GLOBECOM*, pp. 300.3.1–300.3.9, 1990.

28. S. Golestani, "A framing strategy for congestion management," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1064–1077, Sept. 1991.

29. D. Verma, D. Ferrari, and H. Zhang, "Guaranteeing delay jitter bounds in packet switching networks," in *Proceedings of TRICOMM*, pp. 35–43, Apr. 1991.

30. S. Keshav, "On the efficient implementation of fair queueing," *Internetworking: Research and Experience*, vol. 2, pp. 157–173, Sept. 1991.

31. D. Stilliadis, *Traffic scheduling in packet-switched networks: analysis, design, and implemention*, PhD thesis, University of California, Santa Cruz, 1996.

32. S. J. Golestani, "Congestion-free transmission of real-time traffic in packet networks," in *Proceedings of IEEE INFOCOM*, pp. 527–542, Jun. 1990.

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

34. S. Li and N. Ansari, "Scheduling input-queued switches with QoS features," in *Proceedings of the ICCCN*, pp. 107-112, Oct. 1998.

35. R. Venkateswaran, S. Li, X. Chen, C. S. Raghavendra, and N. Ansari, "Improved vc-merging for multiway commnications in atm networks," in *Proceedings of the ICCCN*, pp. 4–11, Oct. 1998.

36. M. J. Karol, K. Y. Eng, and H. Obara, "Improving the performance of input-queued ATM packet switches," in *Proceedings of IEEE INFOCOM*, pp. 110–115, 1992.

37. J. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proceedings of the ACM SIGCOMM*, pp. 143–156, Aug. 1996.

38. 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," in *Proceedings of IEEE GLOBECOM*, pp. 437–441, Nov. 1996.

39. G. Birkhoff, "Three observations on linear algebra," *Rev. Univ. Nac. Tucumán, Ser. A.*, vol. 5, pp. 147–151, 1946.

40. P. R. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Trans. On Automatic Control*, vol. 40, no. 2, pp. 251–260, Feb. 1995.

41. B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," Tech. Rep. CSL-TR-97-738, Stanford University, Nov. 1997.

42. G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, Massachusetts, 1986.

43. P. Hall, "On representatives of subsets," *J. London Math.*, Soc. 10, pp. 26–30, 1935.

44. R. Venkateswaran, C. S. Raghavendra, X. Chen, and V. Kumar, "Hierarchical Multicast Routing in ATM Networks," in *IEEE Intl. Conf. on Communications*, vol. 3, pp. 1690–1694, Jun. 1996.

45. R. Venkateswaran, C. S. Raghavendra, X. Chen, and V. Kumar, "A Scalable, Dynamic Multicast Routing Algorithm in ATM Networks," in *IEEE Intl. Conf. on Communications*, 1997.

46. R. Venkateswaran, C. S. Raghavendra, X. Chen, and V. Kumar, *Support for Group Multicast in PNNI*, ATM Forum draft 97-0076, 1997.

47. G. Armitage, *Support for Multicast over UNI 3.0/3.1 based ATM Networks*, RFC2022, 1996.

48. E. Gauthier, J.-Y. Le Boudec, and D. Dykeman, *SMART: A Many-to-Many Multicast Protocol for ATM*, ATM Forum draft 96-1295, 1996.

49. M. Grossglauser and K. K. Ramakrishnan, *SEAM: A Scheme for Scalable and Efficient ATM Multipoint-to-Multipoint Communication*, ATM Forum draft 96-1142, 1996.

50. S. Komandur, J. Crowcroft, and D. Mosse, "CRAM: Cell Re-labeling at Merge-Points for ATM Multicast," in *Proceedings of IEEE ICATM*, 1998.

51. I. Widjaja and A. I. Elwalid, *Performance Issues in VC-Merge Capable Switches for IP over ATM*, ATM Forum draft 97-0675, 1997.