INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Université d'Ottawa • University of Ottawa

Switching and Scheduling

in ATM and IP over ATM Networks

Author: Peifang Zhou

Supervisor: Prof. Oliver W. W. Yang

A thesis submitted to

the School of Graduate Studies and Research

in Partial Fulfillment of

the Requirement for the Degree of

Doctor of Philosophy

School of Information Technology and Engineering

University of Ottawa

Ottawa, Ontario, Canada

Abstract

The key to multi-service platform performance with QoS support is to provide appropriate switching and scheduling mechanisms at various contention points in a network. In this thesis, we focus on the two key elements in any traffic scheme: switching and scheduling. We introduce innovative switching architecture and scheduling algorithms designed to keep them in line with network operations. In ATM networks, we propose per-VC queueing architecture and per-VC scheduling. In IP networks, we propose per-flow queueing in access routers and VC-merge in core routers to address the scalability issue. We also introduce per-flow scheduling in tandem to our switching architecture. We conduct performance analysis to evaluate the performance of our proposed switching and scheduling schemes. Simulations are used to validate our analytical results.

Acknowledgements

First and foremost, I wish to convey my deepest and sincerest thanks to my supervisor, Professor Oliver Yang, for his invaluable guidance, inspiration, and support throughout the course of my graduate studies.

I am also grateful to the committee members for their insightful questions and comments.

Last but not least, I would like to express my gratitude to my wife and my parents. Without their love and support, this thesis would have been impossible.

Contents

List of Figures

List of Tables

List of Symbols

i, j, k, l: connection or flow identifiers,

C_L: outgoing link capacity at a node,

N: number of (logical) slots in a frame,

es: extra slots available in a frame,

r_i: slot ration for flow i in a frame,

g_i: grant or unused ration accumulated by connection or flow i,

a_i: absorption or extra slot usage accumulated by connection or flow i,

n_i: number of cells at the beginning of a frame for connection or flow i,

u_i: number of cells left in the queue for connection or flow i after cells have been scheduled according to the grant variable,

bw_i: bandwidth requirement of connection or flow i,

T: frame duration,

T_{max}: maximum frame duration,

HEAD(j, v): HEAD register of the address list for output port j and outbound VC v,

TAIL(j, v): TAIL register of the address list for output port j and outbound VC v,

r: state transition probability from an ON to an OFF state,

s: state transition probability from an OFF to an ON state,

ρ: offered load,

$\boldsymbol{\pi}$: steady-state probability of the number of sources in the ON state,

$\boldsymbol{P_{ij}}$: state transition matrix,

$\Pi(z)$: probability generating function of $\boldsymbol{\pi}$,

a_m: number of arrivals in a queue at the end of the mth time slot,

$A(z)$: probability generating function of a_m,

q_m: number of cells in a queue at the end of the mth time slot,

$Q(z)$: probability generating function of q_m,

b_m: buffer occupancy at the end of the mth slot,

B: buffer size.

Chapter 1

Introduction

1.1 Overview

The international standard body CCITT (now called ITU-T) defines the transfer mode as a technique for transmission, multiplexing, and switching aspects of communication networks. In broadband integrated services digital network (B-ISDN), asynchronous transfer mode (ATM) is the switching and transport mode of choice.

The fixed size of 53-byte for ATM cells reduces the variance of delay making ATM networks suitable for integrated traffic consisting of voice, video, and data. Providing the desired quality of service (QoS) for these various traffic types is much more complex than voice networks or data networks of today. Key QoS parameters include delay, delay jitter, bandwidth allocation, buffer occupancy, etc. Proper traffic control helps ensure efficient and fair flow regulation and resource management within the networks in spite of constantly varying demand. This is particularly important for the data traffic which has very little

predictability and therefore can not reserve resources in advance as in the case of voice networks.

Traffic control is one of the most active areas of research in high-speed networking. With communications bandwidth well into the Gb/s range and with Tb/s on the horizon, many of the assumptions that influenced the design of existing traffic control schemes begin to fall apart. One of the fundamental parameters in networking is the ratio between propagation delay and message transmission time [41]. In Gb/s or Tb/s ATM and IP networks, cell or packet transmission time is so small that the above ratio becomes very large, thereby rendering ineffective many of the schemes that rely on feedback from the network in order to regulate traffic flows. The nature of traffic within the networks is also significantly altered. The amount of data traffic, fueled by the widespread used of World Wide Web (WWW), is poised to surpass the amount of voice traffic. Furthermore, real-time services such as interactive video typically require a bandwidth guarantee from the network. Unlike the traditional voice circuits where the bandwidth requirement was fixed (e.g. 64 Kb/s), these new services have bandwidth requirements which may vary by several orders of magnitude, both from connection to connection and at different times within the same connection. This dynamic, heterogeneous, time-varying network environment with different services and QoS requirements, drives researchers to investigate many new concepts and approaches. The research work carried out in this thesis represents our efforts to contribute to this exciting field.

The key to efficient multi-service platform performance in ATM/IP networks is to provide appropriate queueing and scheduling mechanisms at various contention points in

the networks [23]. In this thesis, we focus on the design and analysis of the switching architecture and scheduling algorithm, the two crucial components in any traffic control schemes. We first present our switching and scheduling mechanisms in the context of ATM networks. We then extend these mechanisms to IP networks, which gather tremendous attentions due to explosive growth of the Internet traffic.

1.2 Literature Review

1.2.1 Traffic classification

ATM networks are expected to offer a wide variety of services with diverse traffic characteristics and multiple levels of quality of services (QoS). ATM Forum, which is actively involved in ATM standards development worldwide, classifies traffic according to the following five ATM-layer service categories [21]:

- CBR: Constant Bit Rate,
- rt-VBR: Real-time Variable Bit Rate,
- nrt-VBR: Non-real-time Variable Bit Rate,
- ABR: Available Bit Rate, and
- UBR: Unspecified Bit Rate.

The CBR service category is used by connections that request a fixed (static) amount of bandwidth, characterized by a peak cell rate (PCR) value that is continuously available

during the connection lifetime. The source may emit cells at or below the PCR at any time, and for any duration (or may be silent). The CBR service provides a constant-bandwidth pipe that can be used to support stringent QoS guarantees on the cell loss rate (CLR), maximum cell transfer delay (CTD), and cell delay variation (CDV).

The real-time VBR service category is intended for time-sensitive applications, (i.e., those requiring tightly constrained delay and delay variation), as would be appropriate for voice and video applications. Sources are expected to transmit at a rate which varies with time. Equivalently, the source can be described as "bursty". Traffic parameters are peak cell rate (PCR), sustainable cell rate (SCR) and maximum burst size (MBS). Cells which are delayed beyond the value specified by CTD are assumed to be of significantly less value to the application. Real-time VBR service may support statistical multiplexing of real-time sources.

The non-real time VBR service category is intended for applications which have bursty traffic characteristics and do not have tight constraints on delay and delay variation. As for rt-VBR, traffic parameters are PCR, SCR and MBS. For those cells which are transferred within the traffic contract, the application expects a low cell loss ratio (CLR). For all cells, it expects a bound on the cell transfer delay (CTD). Non-real time VBR service may support statistical multiplexing of connections.

The available bit rate (ABR) is a service category intended for sources having the ability to reduce or increase their information rate if the network requires them to do so. This allows them to exploit the changes in the ATM layer transfer characteristics (i.e., bandwidth availability) subsequent to connection establishment.

The unspecified bit rate (UBR) service category is a "best effort" service intended for non-critical applications, which do not require tightly constrained delay and delay variation, nor a specified quality of service. UBR sources are expected to transmit non-continuous bursts of cells. UBR service supports a high degree of statistical multiplexing among sources.

ATM is designed with integrated QoS support. On the contrary, IP was designed *not* to distinguish packets. The original design goal of TCP/IP is to provide best-effort service to *every* packet, where all transmissions are considered equal and no delivery guarantee are made. There is no traffic classification schemes at all within the original TCP/IP protocol suite.

1.2.2 Switching

Conceptually, ATM networks are packet-switching networks in which each ATM cell in the networks is transported independently along a fixed route. ATM technology is connection-oriented in that the end-to-end path followed by each connection is pre-established before any cell transfer starts. In simple terms, an ATM switch transports cells from the incoming links to the outgoing links using the routing information in the cell header (Virtual Path Identifier (VPI) and Virtual Connection Identifier (VCI) fields) and information stored in the switch. The primary functionality of an ATM switch is to route cells from an input port to the appropriate output port.

When more than one cell attempts to access an output port simultaneously, a phenomenon called output-port contention occurs. When this happens, only one of the con-

tending cells can be sent to the outgoing link. Other cells may be stored in a buffer temporarily or they are simply dropped by the switch. The second functionality of an ATM switch is to buffer cells destined to the same output port at the same time.

Various ATM switches with different architectures have been proposed in the literature. Most designs are centered around the above two functionalities. In other words, they are port-oriented as opposed to connection-oriented. Following the classification in [68], switch fabrics proposed for ATM networks are divided into three categories:

- shared medium architecture [10, 61, 62],
- shared memory architecture [12, 35], and
- space division architecture [28, 45, 70].

Most switch designs reported in two survey papers on ATM switches [1, 2] fall into one of the above three categories.

Packet switching in an IP network is usually accomplished by software and the term routing is used instead of switching. The Internet is the most widely used IP network. The bulk of the Internet uses a distributed scheme in which the path from the source to the destination is not fixed, and each IP packet travels hop by hop between source and destination. Streamlining this routing process could significantly boost the Internet performance. Several vendors have proposed to overhaul the legacy router architecture [36, 50, 56, 64]. Although the details of their implementations vary, there is one fundamental concept shared by all these proposals: map the route information to short fixed-length

labels, so that next-hop routers can be determined quickly through indexing rather than some type of searching (or matching longest prefixes) [65].

1.2.3 Scheduling

Traffic scheduling is used in diverse environments to satisfy a variety of objectives. A common application of scheduling algorithms is to provide network-level QoS to connections by isolating their traffic from other, possibly misbehaving, traffic. Another application is to achieve statistical multiplexing gain by enabling connections to share any available bandwidth in an equal or weighted fashion.

The literature is voluminous in the research area of fair scheduling in packet and ATM networks. In recent years, many ingenious algorithms have been proposed, such as virtual clock [72], generalized processor sharing [53], self-clocked fair queueing [24], virtual spacing [57], WF2Q [3], logarithmic calendar queue [9], etc. Most algorithms are based on the idea of associating to each arriving packet, or cell in the context of ATM, a virtual time-stamp calculated with respect to an increasing reference virtual clock. The smaller the time-stamp, the higher the priority of the corresponding packet or cell to access the transmission facility. These algorithms proved efficient in guaranteeing the fairness among traffic flows.

1.3 Motivations and Objectives

1.3.1 Motivations

Switching and scheduling are key to enable QoS in any traffic control schemes in ATM and IP networks. Therefore, we shall focus on these two crucial components in this thesis. In this section, we shall make an observation on the shortcomings of current switching and scheduling schemes as discussed in the Literature Review section from which we shall propose our remedies.

Traffic classification

ATM Forum defines five service categories (CBR, rt-VBR, nrt-VBR, ABR, UBR) as a way of classifying traffic carried within ATM networks. The above classification method is too complicated, and ATM Forum is still struggling on how to deal with bursty data transfer. Originally, only one type of VBR service was defined. As time evolves, it was recognized that we needed to distinguish between real-time video traffic and non-real-time data traffic. So VBR was split into two flavors: rt-VBR and nrt-VBR. This improved bursty data traffic handling, but it did not solve the problem completely. ATM Forum introduced ABR service to exploit idle bandwidth for data bursts. But the rate-based approach is too complicated and full-fledged ABR flow control has not been supported in mainstream ATM switches yet. From ATM end-users' perspective, they can not predict their traffic pattern, and therefore they can not inform the underlying ATM network about all those traffic parameters such as PCR, SCR, MBS, etc. The nrt-VBR or ABR service is

just too complex and expensive for bursty data transfer. ATM Forum is now considering to introduce another new service category called UBR+ or Guaranteed Frame Rate (GFR) [60], which is designed to address ATM end-users' requirement of choosing a really simple service like UBR, while receiving a minimum throughput guarantee from the ATM network. At the time of this writing, specification on GFR is not finalized yet. There is much and continuous interest in traffic classification for simple but efficient network operation.

Switching

Traditional ATM switch designs [1, 2] were port-oriented. They were based on "first-in, first-out" (FIFO) principle: cells leaving an output port were organized into a FIFO queue and processed in the order in which they arrived. The FIFO queueing worked well for homogeneous traffic, but failed badly for heterogeneous traffic. The drawback of FIFO queueing has long been recognized, and many variations of FIFO queueing (e.g. weighted fair queueing [15]) have been proposed. For any connection-oriented technology such as ATM, we feel that the answer to an effective solution lies in having a separate queue for each connection.

Scheduling

The scheduling algorithms discussed in the Literature Review section are based on the idea of associating to each arriving packet, or cell in the context of ATM, a virtual time-stamp calculated with respect to an increasing reference virtual clock. The smaller the time-stamp, the higher the priority of the corresponding packet or cell to access the transmission

facility. These algorithms proved efficient in guaranteeing the fairness among traffic flows. However, the inherent complexity of the virtual time-stamp approach makes practical implementation difficult and expensive, especially in high-speed Tb/s networks. Moreover, the use of a reference clock implies that it can not be reset to zero until the system is idle. Research is still much needed in the area of scheduling to produce feasible algorithms in Tb/s networks, while achieving bandwidth guarantee and fair allocation at the same time.

1.3.2 Objectives

The general objective of our research is to seek alternatives in traffic classification, switching, and scheduling in order to address the shortcomings observed in the previous subsection of Motivation. We shall focus on switching and scheduling which are key to enable QoS in any traffic control schemes.

To be more specific, we shall

- propose a new traffic classification scheme in which we shall avoid the complexity in the traffic classification and come up with a simple and effective scheme,
- introduce a new switching architecture in line with network operations; In this architecture, we shall break the traditional port-oriented approach in the ATM switch design and apply a connection-oriented approach,
- present a new scheduling mechanism in which we shall do away with any cell-based methods and use a connection-oriented technique to take full advantage of connection-oriented switching architecture, and

- explore the possibility of extending our proposed switching and scheduling schemes to high-speed IP networks.

1.4 Methodology and Approaches

In this section, we detail how we fulfill our general objective of addressing various issues related to switching and scheduling and introducing innovative solutions in terms of switching architecture and scheduling mechanism.

We use a network model which is widely implemented in practice, such as the Internet. In this network model, nodes/switches are interconnected to form a network. Hosts/terminals are attached to switches at the network edge. This commonly used network model can be found in any textbook on networking, e.g. [58].

For traffic characterization, we apply popular models for various types of traffic. We adopt the use of constant bit-rate model for voice traffic, variable bit-rate model for video traffic, and ON-OFF model for bursty data traffic. The above traffic models can be easily found in the literature, e.g. [58]. For queueing models used in performance analysis, the backbone network cloud is treated as tandem queues.

In our work on traffic classification, we do not follow ATM Forum which defines a large number of traffic classes and complex traffic descriptors. Our proposal is to introduce a small number of classes and do away with complex traffic descriptors in both ATM and IP networks. The benefits of such a scheme include realistic specification on bursty traffic by end users and simple network operations for service providers.

In our work on designing ATM switches, we do not take a port-oriented approach which has been prevalent in the literature. Port-oriented approaches limit the design focus only on the buffering and routing aspect of an ATM switch. In our research, we take a connection-oriented approach to keep the ATM switch design in synchronization with network operation, which is to provide transport services on an end-to-end basis. With this design goal in mind, we come up with an ATM switch design with per-VC (Virtual Connection) queueing architecture. To the best of our knowledge, our paper on per-VC queueing design [77] was the first in the literature. To evaluate the performance of per-VC queueing architecture, queueing analysis is carried out to analyze the buffer occupancy distribution.

Per-VC queueing architecture is powerful in the sense that it provides isolation to each VC so that precise traffic management can be exerted on each individual connection without affecting other VCs. The shift in the structure of ATM switches is significant, and it has profound implications on the operations of ATM networks. One of these implications is its impact on scheduling.

In our work on designing a scheduling algorithm for per-VC queueing ATM switches, we do not take a per-cell based approach which has been dominant in the literature. Per-cell based method is cost-prohibitive in high-speed Tb/s networks. Instead, we move up one level of "abstraction" and take a connection-oriented approach, which significantly reduces the computational effort and is also in line with network operation of providing end-to-end transport services. We introduce a Per-VC Queueing Scheduler (PVQS) which handles scheduling in the output port of per-VC queueing ATM switches. Since our scheduling

computation is carried out on a per connection (per flow) basis, not on a per cell (per packet) basis, we demonstrate that this leads to a significant reduction in the computational effort, and makes the algorithm suitable for practical implementation in high-speed networks. To evaluate the performance of PVQS, tandem queueing model is used and queueing analysis is carried out to derive the end-to-end delay and delay jitter bounds.

After the study of switching and scheduling in the ATM networks, we explore the potential of extending our schemes into IP networks. TCP/IP was originally designed to support best-effort services. With the introduction and explosion of World Wide Web, the challenge of providing QoS support in IP networks starts to attract the attention of researchers. We find that switching and scheduling are still key to enable QoS in IP networks. On the design of IP switching routers, we use the per-VC queuing architecture as the foundation and conceive integrated cell and packet switching in routers to switch ATM cells and IP packets simultaneously. We also introduce modifications to the per-VC queueing architecture to address the concern of scalability for backbone routers. On the design of scheduling algorithms, we extend the idea of PVQS and introduce per-flow scheduling (PFS) as the scheduling mechanism in IP switching routers. To evaluate the performance of PFS, tandem queueing model is used and queueing analysis is carried out to analyze end-to-end delay, delay jitter, throughput, and buffer occupancy distribution.

To verify the validity of our analysis, simulations are conducted and results obtained are used to compare with analytical ones. Due to lack of support for our switching and scheduling models in commercial simulation packages such as OPNET [52], we have to resort to C or C++ programming language to write our own simulation code.

1.5 Contributions and Thesis Organization

The contributions of this thesis are listed as follows:

- a new traffic classification scheme based on the delay attribute; This scheme defines a small number (3 to be specific) of traffic classes to facilitate marking at the edge and merging in the core,
- a new type of switching architecture which is connection-oriented as opposed to traditional port-oriented design; Unlike other switches [27] with completely shared buffering, our proposed per-VC queueing architecture does *not* require an increase of the switch size. To the best of our knowledge, our paper on per-VC queueing architecture [77] was the first one reported in the literature,
- a new per-VC based scheduling method for use with the per-VC queueing; This is different from a per-cell based approach,
- a new per-flow queueing architecture in IP networks; This architecture allows us to integrate frame and cell switching in access routers and VC-merging in core routers;
- a new design of VC-merging in the core routers with a simple addition/modification to the per-VC queueing ATM switches,
- a new per-flow based scheduling mechanism instead of per-packet based technique, and

- performance analysis of our proposed switching architecture and scheduling algorithms that would demonstrate the validity of our design.

The rest of the thesis is organized as follows. Chapter 2 presents the network and traffic models. Chapter 3 studies switching and scheduling in ATM networks. We shall present per-VC queueing architecture in the ATM switch design and per-VC based scheduling. We shall carry out performance analysis to verify our design. In Chapter 4, we extend the above mentioned switching and scheduling schemes to IP networks. We shall present integrated frame and cell switching and per-flow based scheduling to provide QoS support in IP networks. We shall conduct performance analysis to validate our design. In Chapter 5, we provide some design guidelines. Chapter 6 concludes the thesis.

1.6 Publication List

The following is a list of our published papers:

- P. Zhou and O. Yang, "Design and analysis of per-flow queueing switches and VC-merge switches based on the per-VC queueing architecture," *Computer Commun.*, special issue on support of multimedia communications over the Internet, vol. 23, no. 14-15, pp. 1400-1409, Aug. 2000.
- S. Jha, P. Zhou, and O. Yang, "Performance evaluation of joint per-VC scheduling and credit-based flow control in high-speed ATM networks," in *Proc. IEEE IPCCC'00*, Phoenix, AZ, Feb. 2000, pp. 571-577.
- P. Zhou and O. Yang, "Reducing buffer requirement for VC-merge capable ATM switches," in *Proc. IEEE GLOBECOM'99*, Rio de Janeiro, Brazil, Dec.1999, pp. 44-48.
- P. Zhou and O. Yang, "Scalability and QoS guarantee in IP networks," in *Proc. IEEE IC3N'99*, Boston, MA, Oct. 1999, pp. 427-433.

- P. Zhou and O. Yang, "Queueing and scheduling in the design of QoS-capable routers," in *Proc. SPIE Performance and Control of Network Systems*, Boston, MA, Oct. 1999, pp. 49-60.
- P. Zhou and O. Yang, "Integrated cell and frame switching in ATM networks," *IEEE Commun. Lets.*, vol. 3, no. 6, pp. 183-184, Jun. 1999.
- P. Zhou and O. Yang, "A feasible scheduling algorithm for per-VC queueing ATM switches," in *Proc. ICATM'99*, Colmar, France, Jun. 1999, pp. 295-304.
- P. Zhou and O. Yang, "Design and analysis of per-flow queueing switches and VC-merge capable switches based on per-VC queueing architecture," in *Proc. IEEE BSS'99*, Kingston, Ontario, Canada, Jun. 1999, pp. 19-23.
- P. Zhou and O. Yang, "Reverse routing: an alternative to MIP and ROMIP protocols," in *Proc. CCECE'99*, Edmonton, Alberta, Canada, May 1999, pp. 150-155.
- P. Zhou and O. Yang, "A framework of flow control in high-speed ATM networks," in *Proc. IEEE MILCOM'98*, Bedford, MA, Oct. 1998, pp. 782-786.
- P. Zhou and O. Yang, "Traffic control in high-speed ATM networks," in *Proc. IC3N'98*, Lafayette, LA, Oct. 1998, pp. 183-190.
- P. Zhou and O. Yang, "Design of per-VC queueing ATM switches," in *Proc. IEEE ICC'98*, Atlanta, GA, Jun. 1998, pp. 304-308.
- P. Zhou and O. Yang, "Flow control in high-speed ATM wide area networks," in *Proc. CCECE'98*, Waterloo, Ontario, Canada, May 1998, pp. 249-252.
- P. Zhou and O. Yang, "Cell loss analysis of per-VC queueing ATM switches," in *Proc. CCECE'98*, Waterloo, Ontario, Canada, May 1998, pp. 245-248.
- P. Zhou and O. Yang, "A new design of central-queueing ATM switches," in *Proc. IEEE GLOBECOM'97*, Phoenix, AZ, Nov. 1997, pp. 541-545.
- P. Zhou and O. Yang, "A buffering scheme in ATM switches," presented at *CORS'97*, Ottawa, Ontario, Canada, May, 1997.
- P. Zhou and O. Yang, "A novel architecture design of ATM switches," in *Proc. CISS'97*, Baltimore, Maryland, Mar. 1997, pp. 57-62.
- P. Zhou and O. Yang, "Delay minimization in a multicasting tree," *IEICE Trans. on Commun.*, vol. E80-B, no. 2, pp. 301-306, Feb. 1997.
- P. Zhou and O. Yang, "Delay minimization in a multicasting tree," presented at *IEEE Symposium on Planning and Design of Broadband Networks*, Montebello, Quebec, Canada, Oct. 1996.

• P. Zhou and O. Yang, "Finding branching points in a multicasting tree," in *Proc. CISS'96*, Princeton, NJ, Mar. 1996, pp. 717-722.

Chapter 2

Network Description and Models

In this chapter, we discuss the network model and the traffic model to be used in the rest of this thesis.

2.1 Generic Layout of Network under Consideration

Figure 2.1 illustrates a generic topology for an ATM (IP) network in our investigation: users/hosts are attached to access switches (routers), and access switches (routers) are interconnected to a backbone cloud formed by core switches (routers). The job of the access/core switches (routers) is to carry data units (cells in ATM networks and packets in IP networks) from host to host to enable end-to-end communications. By separating the pure communication aspects of the network (the access and core switches/routers) from the application aspects of the network (the hosts), the complete network design is greatly simplified [63]. This commonly used network model can be found in any textbook on

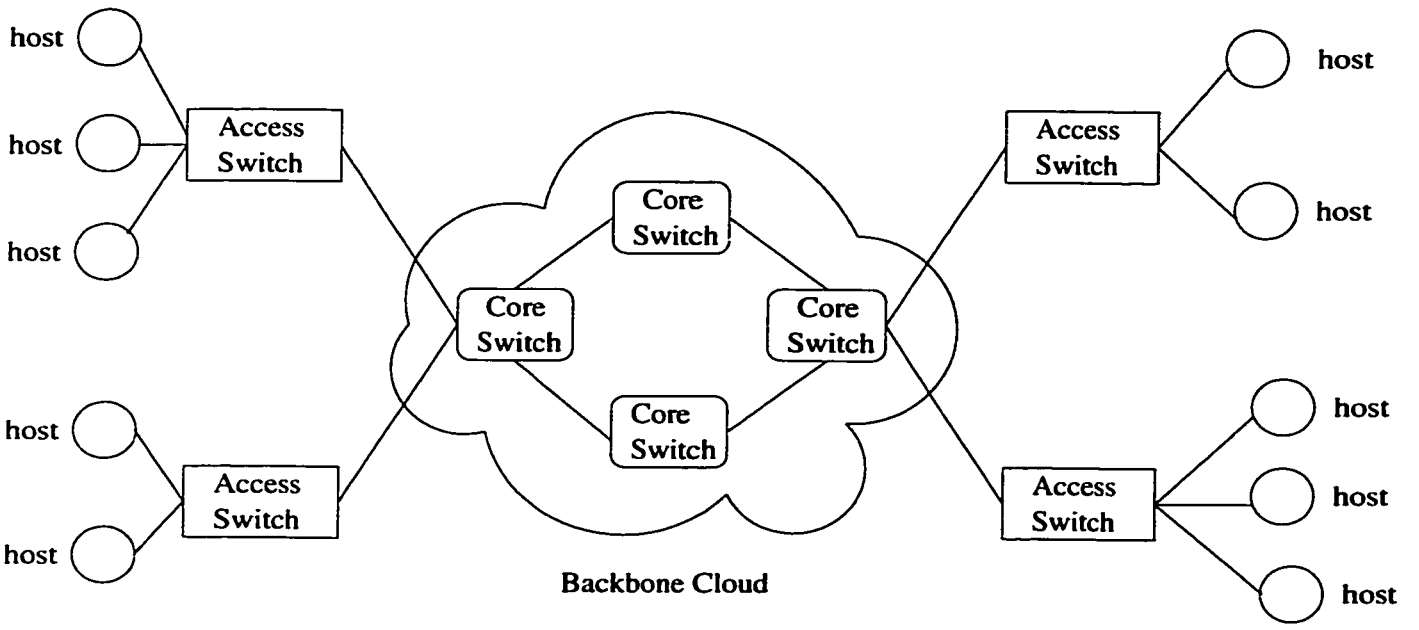

Figure 2.1: Generic network model.

networking, e.g. [58, 63].

2.2 Traffic Models

In this thesis, we apply popular models for various types of traffic. We adopt the use of constant bit-rate model for voice traffic, variable bit-rate model for video traffic, and ON-OFF model for bursty data traffic. The above traffic models can be easily found in the literature, e.g. [58].

It has been known for many years that an analog voice signal can be sampled and quantized at a fixed rate, resulting in a 64 Kb/s stream traffic. Lower bit-rates are also feasible for voice traffic with proper encoding techniques such as ADPCM. It is natural to

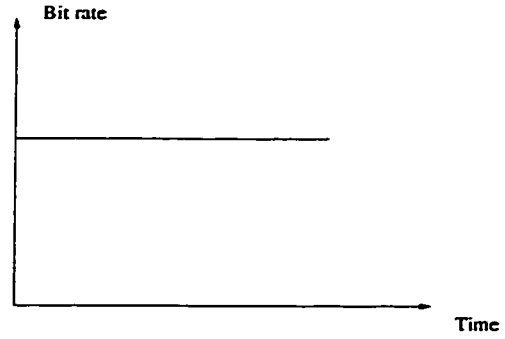

Figure 2.2: Constant bit-rate traffic.

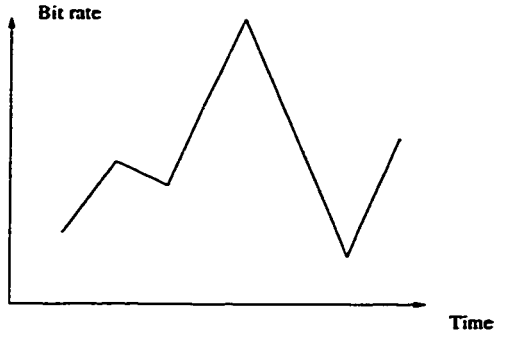

Figure 2.3: Variable bit-rate traffic.

use a constant bit-rate model to characterize voice-type sources which generate traffic at a constant bit-rate. Figure 2.2 shows the bit-rate of a voice traffic source as a function of time. If silence suppression is used to reduce the bandwidth consumption, voice traffic may exhibit variable bit-rate.

In its raw, uncompressed format, a video signal is representative of constant bit-rate traffic. Compression techniques such as MPEG [31] remove inherent redundancy in the video signal and the compressed video stream traffic has variable bit-rate over time, which is represented schematically in Figure 2.3. MPEG video traffic can also be constant bit-rate

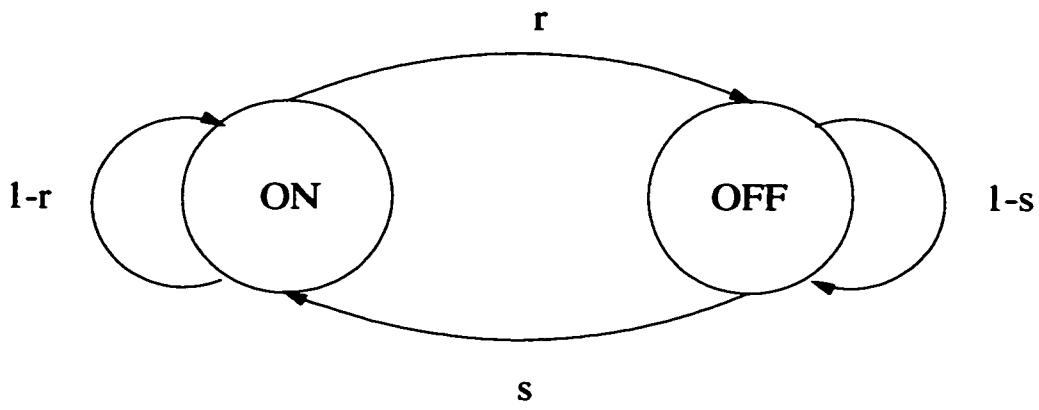

Figure 2.4: ON/OFF model for bursty data traffic.

if adaptive encoding is employed.

It is also well known that data traffic such as TCP/IP traffic is bursty, which can be characterized by short periods of activity interspersed with long idle periods. To represent burstiness of data traffic, we use the so-called ON-OFF model in which a bursty traffic source alternates between ON and OFF states. In the OFF state, the source does not generate any ATM cells. In the ON state, the source emits data cells at a fixed rate. There is a state transition probability associated with each state. Figure 2.4 portrays the above ON-OFF model.

The Markov Modulated Poisson Process (MMPP) [13, 16] has been widely used to model various traffic sources, such as voice and video, as well as characterizing the superposed traffic. However, traffic measurements in both Local Area Networks (LANs) and Wide Area Networks (WANs) show that traffic patterns do *not* follow the Poisson model. In fact, both LAN and WAN traffic have a self-similar or fractal characteristic [46, 54]. It is suggested in [67] that aggregating a large number of ON-OFF sources would give rise

to a self-similar process. Based on the above theoretical result, a conjecture is made in [69] that ON-OFF behavior provides the physical basis for the self-similarity observed in packet data traffic. It is the main reason why we choose the ON-OFF model for bursty traffic. In other words, each bursty traffic source is modeled as an ON-OFF source, while aggregation of a large number of such ON-OFF sources may lead to a self-similar traffic pattern observed at the network level.

2.3 Traffic Classification

Even though there can be many types of traffic according to their applications, in general they can be grouped/classified by their characteristics. Our key idea in traffic classification is to define a small number of traffic classes to facilitate marking at the edge and aggregation in the core. This idea was first conceived in the context of ATM networks [74] and it was further extended to IP networks [80].

ATM allows transport of mixed traffic streams and there are essentially two types of traffic which require QoS guarantee within ATM networks. One type is continuous-stream traffic such as voice and video, which are sensitive to delay but tolerant to a certain level of cell loss. Another type is bursty data traffic such as IP flows, which are sensitive to cell loss but can accommodate delay. With a well-tuned credit-based flow control, e.g. [78], zero cell loss can be guaranteed. The reason is the following. Credits represent the amount of buffer space available at the next-hop receiver. If the transmitter does not exceed this quantity of data, there is no risk of buffer overflow. Even under extreme overloads, queue

lengths inside switches can not grow beyond what the credits allowed. Note that credit-based flow control ensures zero cell loss within a network while cell loss can happen at the edge of a network if call admission control and policing (e.g., leaky-bucket) are not properly enforced. Flow control is outside the scope of this thesis.

Since we can eliminate cell loss for data traffic, we can classify traffic based only on the delay attribute to distinguish real-time and data traffic. In our classification scheme, we divide traffic into three types. ATM and IP networks will provide delay-sensitive class of service and delay-insensitive class of service with QoS guarantee, while keeping the best-effort class of service without QoS support for backward compatibility.

Although our proposed method simplifies traffic classification, it has drawbacks in comparison with ATM Forum's approach. The granularity of ATM Forum's approach is smaller and more fine-tuned. As a result, ATM Forum's scheme is more flexible and easier to accommodate various combinations of traffic requirements. For example, a user can choose rt-VBR service category to transport a MPEG video sequence which is bursty and has a real-time requirement. If the same user has to use our classification method, he/she can choose delay-sensitive service category to satisfy the real-time requirement. For a delay-sensitive traffic, peak-rate bandwidth requirement needs to be specified in order to obtain delay and delay jitter guarantees. This means that the user has to translate the maximum burst size (MBS) in the MPEG video sequence into a bandwidth requirement, and use that as the peak-rate specification. This would result in a higher bandwidth specification when compared with the ATM Forum's rt-VBR case. Due to this higher bandwidth requirement, it is possible that a connection request is rejected while it would be accepted if

ATM Forum's method is employed.

There exist cases, namely traffic associated with routing protocols and traffic related to transactions, in which only a few packets are exchanged (we call them short transfers). One way to handle these short transfers is to treat them as a special flow and reserve resources accordingly. This strategy will work well as long as short transfers compose a small portion of the total traffic, and the aggregation remains stable [71].

2.4 General Assumptions

There are two general assumptions carried out in the rest of this thesis. First, the Dense Wavelength Division Multiplexing (DWDM) technology will provide an extremely high transport capacity in the order of 10 Tb/s to address the concern of explosive traffic growth, so that we have abundant bandwidth available in the backbone of an ATM/IP network. Secondly, both real-time (e.g. voice) and data traffic would be carried by a common public switched network. It is observed that data traffic expands exponentially, while voice traffic grows linearly. Therefore voice would become a small fraction of the total traffic. It is not difficult to justify the assumption that data traffic would be dominant in the next-generation networks. In fact, statistics have shown that data traffic has taken over voice [11, 30].

To maintain clarity of thesis presentation, we choose to ignore the best-effort class of traffic in the rest of this thesis.

Chapter 3

Switching and Scheduling in ATM Networks

There are many aspects of traffic control, and we focus on the two key characteristics in any traffic control scheme: switching and scheduling. On the switching side, we present a new per-VC queueing architecture as the cornerstone in our ATM switch design. Per-VC queueing also simplifies the design of scheduling since a scheduler can obtain information for every connection, which leads to the development of a Per-VC Queueing Scheduler (PVQS) in this thesis.

3.1 Switch Architecture

We follow the classification scheme used in [27] and divide the buffering (queueing) strategies according to the physical location of the buffers: at the inputs, at the outputs, or

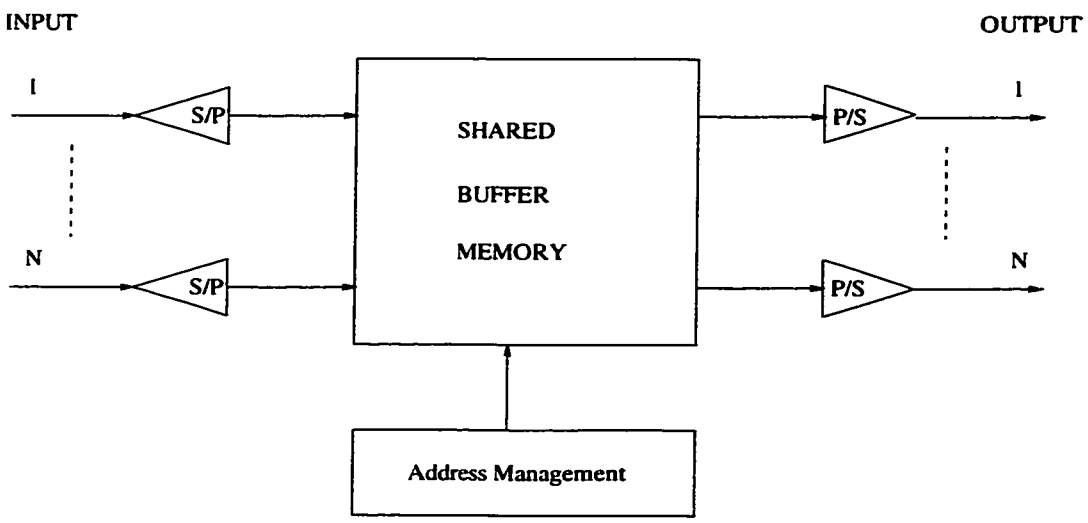

Figure 3.1: Central queueing switch architecture.

inside the switch (central queueing). Central queueing discipline avoids the head-of-line (HOL) blocking of input queueing. It achieves the optimal throughput-delay performance of output queueing [27], as well as realizing a better utilization of buffer space than output queueing for a desired cell loss probability. Therefore we choose central queueing discipline for our proposed switch architecture as shown in Figure 3.1. When cells enter the switch, they are first converted from serial to parallel (S/P) to reduce the internal operation speed of the switch. Note that there is a reverse operation from parallel to serial (P/S) before cells leave the switch.

The central buffer memory is shared among all virtual connections. The operation of the switch is time-slotted based on a cell time. In one time slot, cells from N input ports are first written into the buffer memory and cells are then read out of the buffer memory to N output ports.

The basic function of an ATM switch is to route cells from an input port to an output

port, and to buffer cells destined to the same output port. In our proposed ATM switch architecture, we do not implement the routing functionality of the ATM switch by moving cells through the usual sorting and routing circuits [14, 2]. Instead we use an innovative scheme, shown in Figure 3.2, which stores all ATM cells *statically* in the buffer and combines the functionalities of routing, buffering, and separation of traffic flows by the *per-VC queueing address management*. Unlike other switches [27] with completely shared buffering, the proposed switch architecture does *not* require an increase of the switch size.

All the intelligence of our proposed ATM switch architecture resides in the Per-VC Queueing Address Management module, shown in Figure 3.2. We present a detailed look at its architecture in this section.

The Per-VC Queueing Address Management module consists of four parts: an Address Token Pool, a Reference Number module, M address lists for M virtual connections, and N Scheduling modules for N output ports. Of those M address lists, M_1 are associated with output port 1, M_2 with output port 2, ..., M_N with output port N, where $M_1 + M_2 + \ldots + M_N = M$. The number of address lists associated with each output port is configurable and can be changed on the fly. Therefore our proposed architecture is flexible. There are two registers, HEAD register and TAIL register, for each address list. Those registers allow a fast and easy access to the corresponding address list.

A token represents a cell address in the shared buffer. There is a one-to-one mapping between token and cell address in the shared buffer. There is also a reference number for each cell address (token). The Reference number is used to keep track of the number of destinations for each received cell. When a unicast cell is received from an input port, the

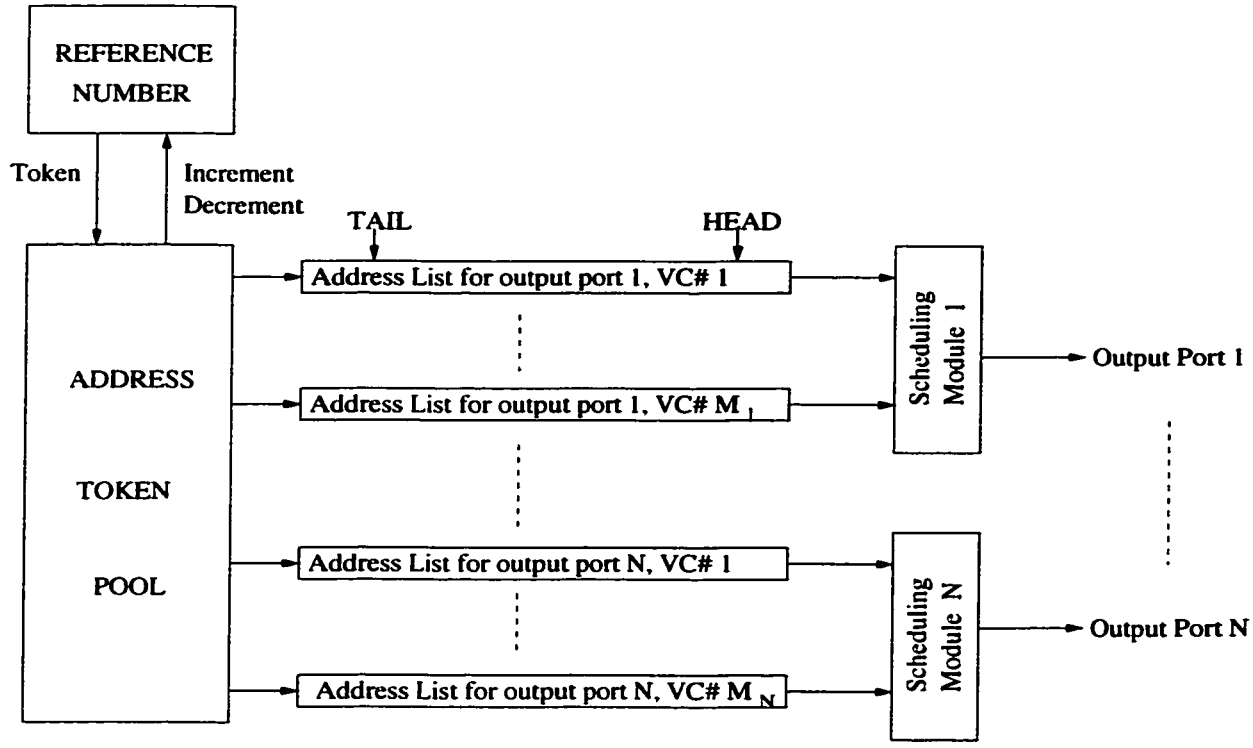

Figure 3.2: Per-VC queueing address management.

Address Token Pool will release a token and place it in the address list for the corresponding output port and outbound VC. The reference number for the selected cell address (token) will be increased from 0 to 1. The received cell will be written at the specified cell address (token). If a multicast cell is received, the Address Token Pool would still release only *one* token, but it would make multiple copies of the released token and put them in the multiple address lists for the corresponding outbound VCs and output ports which the received multicast cell are destined to. The reference number of the cell address (token) will be increased from 0 to the number of destinations of the multicast cell.

The Per-VC Queueing Address Management module uses the address lists to check if a cell needs to be read out of the shared buffer and sent to the corresponding output port. If the address lists associated with an output port are not empty, the Scheduling module for the output port will select one of the non-empty address lists, use the cell address (token) pointed by the HEAD register to read the cell out of the shared buffer, and send it to the corresponding output port. The reference number for the cell address (token) is decremented by 1. The HEAD register then adjusts and points to the next token in the address list. When the reference number of a token reaches zero, a new token will be generated and it will replenish the Address Token Pool.

There are many ways to implement the Address Token Pool module. One of them is to use the method of linked list, which collects the address of cells which have been read out of the shared buffer memory and sent to the output port. This linked list can be called "free list". In the next section, we use the above linked list technique to implement the Address Token Pool module.

3.2 Switch Operation

The linked list operation supports naturally two types of functions common in nowadays networks: unicast and multicast.

3.2.1 Unicast

The operation of an $N \times N$ switch can be shown by the following manipulation of the $M + 1$ linked lists mentioned in the previous section. Of those $M + 1$ linked lists, M of them are used to support all the M virtual connections allowed in the switch. We use the notation list(j, v) to denote the address list associated with output port j and outbound VC number v. The last linked list functions as the Address Token Pool and it is called "free list".

The operations of our per-VC queueing switch are detailed as follows.

Step 1: Initialization. The free list is initialized with the whole address space of the buffer memory, i.e., every cell address (token) is on the free list.

Step 2: In one cell-time slot, cells from N input ports, if any, are written into the buffer memory. For every input port $i, i = 1, \ldots, N$, the $(M+1)$st linked list (free list) is updated as follows. The cell which is received from input port i and destined to output port j with an outbound VC number v will be written into the address indicated by the content of HEAD$(M + 1)$ register of the free list. The address list(j, v) is appended with this cell address, and TAIL(j, v) is adjusted to reflect the fact. HEAD$(M + 1)$ then moves on and points to the next element in the free list.

Step 3: In the same slot, cells will be read out of the buffer memory and sent to the corresponding output ports. For every output port $j, j = 1, \ldots, N$, the Scheduling module j selects one of the M_j address lists, uses the content of HEAD(j, v) to read out a cell, and sends it to the output port j with VC number v. The content of HEAD(j, v) is added to the free list by adjusting TAIL$(M + 1)$. HEAD(j, v) then moves on and points to the next elements of the address list(j, v).

Step 4: Repeat the above Step 2 and Step 3 for the next cell-time slot.

In the above algorithm, the linked list manipulation is simple and it lends itself readily to a straightforward VLSI implementation [29].

3.2.2 Multicast/broadcast and priority scheduling

Multicast/Broadcast is one of the main functionalities that should be supported by ATM switches. Multicast/Broadcast can be easily supported by our new address management scheme, without duplicating copies of a multicast cell inside the buffer memory. Only minor modifications to the above unicast algorithm are needed. When a multicast cell is written into the buffer memory, a reference number should also be stored as an indication of the number of multicast output ports. All TAIL registers of the linked lists associated with those multicast output ports and outbound VCs will be updated, following the same procedure outlined in the above unicast algorithm. When a multicast cell is read out, the reference number is decreased. The TAIL$(M + 1)$ register of the free list is updated when the reference number reaches zero. All other operations remain the same as before.

Priority scheduling is another important functionality which should be supported by

ATM switches. As in the case of Multicast/Broadcast, priority scheduling can be easily supported by our new address management scheme. As shown in Figure 3.2, our design of per-VC queueing architecture separates traffic flows. The Scheduling module for output port j can access any one of the M_j address lists. Scheduling algorithms can be programmed/downloaded to Scheduling modules on a per port basis. This architecture lends itself to a great flexibility of selecting a variety of scheduling policies for every output port.

3.3 Per-VC Queueing Scheduler (PVQS)

In this section, we introduce a scheduling algorithm for per-VC queueing ATM switches.

3.3.1 Model description

In the previous subsection, we describe an architecture of per-VC queueing. In a nutshell, per-VC queueing allocates a separate queue for each virtual connection, i.e., there will be M queues if an output port of ATM switch supports M virtual connections.

As shown in Figure 3.3, the per-VC queueing system under study consists of a single server of capacity C_L (link capacity), one queue for each of the M virtual connections. Each virtual connection generates ATM cells into its corresponding queue. The outgoing link is cell-based and time-slotted, with each time-slot being the transmission time of one ATM cell. A link scheduling policy dictates which ATM cell to transmit next.

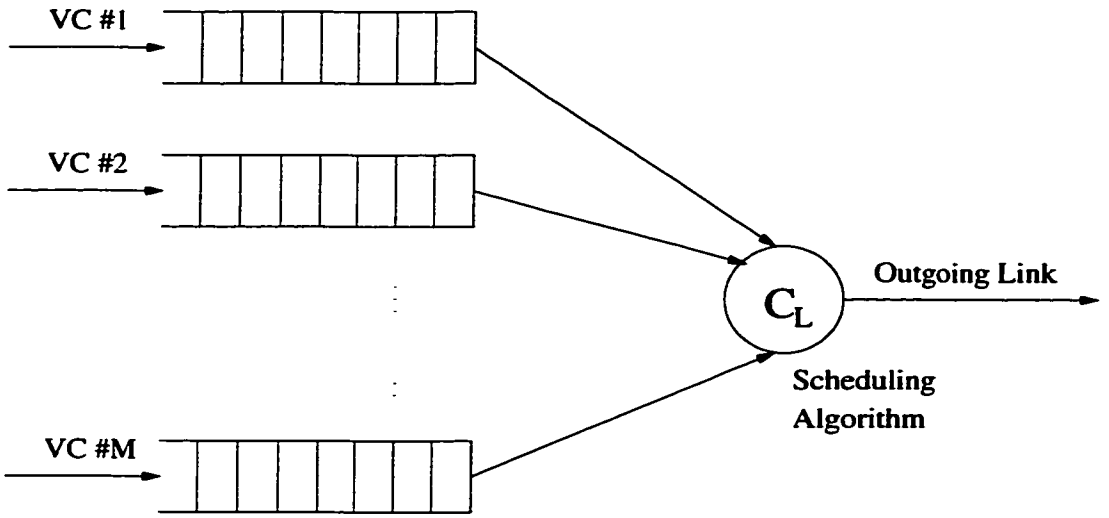

Figure 3.3: Per-VC queueing system.

3.3.2 Operations of PVQS

In this section the following notations pertain:
i, j, k, l: connection identifiers,
C_L: outgoing link capacity at a node,
N: number of (logical) slots in a frame,
es: extra slots available in a frame,
r_i: slot ration for connection i in a frame,
g_i: grant or unused ration accumulated by connection i,
a_i: absorption or extra slot usage accumulated by connection i,
n_i: number of cells at the beginning of a frame for connection i,
u_i: number of cells left in the queue for connection i after cells have been scheduled according to the grant variable,
bw_i: bandwidth requirement of connection i.

Per-VC queueing scheduler (PVQS) resides in every outgoing port. PVQS selects cells for transmission on a frame basis (logically) with N slots in a frame. It will be shown later that an introduction of an N-slot frame structure guarantees not only a bandwidth, but also a low maximum delay in comparison with the end-to-end propagation delay.

To provide bandwidth guarantee for each connection, the idea of "slot ration" needs to be introduced. If the link capacity is C_L and connection i requires a bandwidth of bw_i, the slot ration r_i for connection i in a frame is

$$r_i = \frac{bw_i}{C_L} \times N. \tag{3.1}$$

At the beginning of a frame, PVQS inspects all VC queues associated with the link, and schedules transmission of cells in each VC queue up to the VC's ration. Virtual connections are divided into two sets: UR and OR. UR designates the set of connections for which the numbers of cells at the beginning of a frame are under their rations, i.e.,

$$UR = \{\text{connection } k \mid n_k \leq r_k\}, \tag{3.2}$$

where n_k is the number of cells in the VC queue for connection k at the beginning of a frame, r_k is the slot ration for connection k.

OR represents the set of connections for which the numbers of cells at the beginning of a frame are over their rations, i.e.,

$$OR = \{\text{connection } l \mid n_l > r_l\}, \tag{3.3}$$

where n_l is the number of cells in the VC queue for connection l at the beginning of a frame, r_l is the slot ration for connection l.

From the scheduling point of view, some slots in an N-slot frame may be un-allocated,

and not all slots allocated to connections in the set UR are utilized. We can calculate unused extra slots es as follows:

$$es = (N - \sum_i r_i) + \sum_k (r_k - n_k),$$
$$i \in \{UR, OR\}, \quad k \in UR, \tag{3.4}$$

where the first term represents un-allocated slots in a frame, and the second term is the number of un-utilized slots left by connections in the set UR.

To achieve high link utilization and multiplexing gain among connections, we need to re-allocate these extra slots, and use them to dispatch cells which belong to connections in the set OR. In the mean time, connections in UR should be credited for "freeing up" slots which allows the scheduler to transmit cells from OR connections. UR connections should be compensated by being allowed to transmit more cells than rations later on. It is well known that data traffic are bursty. When a burst arrives, a connection may have to make the transition from the set UR to the set OR, and it is fair for the connection to claim previously unused rations in order to transmit more cells and get the burst quickly out of the VC queue. In other words, the scheduler should schedule more cell transmissions for connections which have not used up their rations. This should be viewed as part of the scheduler's commitment of providing bandwidth guarantee to every connection. For this purpose, we introduce a *grant* variable, g_i for connection i, to keep track of unused ration on a per-VC basis. The value of g_i is non-negative with an initial value of 0 (zero). It is incremented when fewer cells than connection i's ration are scheduled at the beginning of

a frame, i.e., the updated value $\hat{g}_k$ is

$$\hat{g}_k = g_k + (r_k - n_k), \quad n_k < r_k, \tag{3.5}$$

and it is decremented when more cells (up to g_k) than connection i's ration are scheduled during a frame,

$$\hat{g}_k = g_k - m_k, \quad m_k < g_k, \tag{3.6}$$

where m_k is the number of cells transmitted over connection k's ration.

PVQS calculates extra slots available at the beginning of each frame, and selects OR connections which have *grants* available for transmitting more cells than their rations. The selection is based on the *grant* variable. The larger the value of a *grant* variable, the higher priority a connection has. If *grants* are used up for all connections but extra slots are still not exhausted, the scheduler will select connections in the set OR for cell transmission. Based on the assumption that end users pay more for higher bandwidth, the scheduler will distribute extra slots based on the bandwidth requirement. The higher bandwidth a connection requires, the higher priority the connection has. Here we need to introduce another *absorption* variable to keep track of extra bandwidth consumption on a per-VC basis. The *absorption* variable, a_i for connection i, is non-negative. It starts at 0 (zero), and it is incremented when a connection takes advantage of extra slots available in a frame to transmit more cells. The smaller the value of a *absorption* variable, the higher priority a connection has in consuming any extra slots. To maintain fairness among connections which have requested different bandwidths and to reflect tariff imposed on different rates,

any extra bandwidth left by connections in the set UR should be distributed in proportion to bandwidth requirements for connections in the set OR. In other words, extra slot usage needs to be normalized by the bandwidth requirement. If m_l extra slots are used by connection l, its absorption variable a_l should be updated as follows:

$$\hat{a}_l = a_l + K \times \frac{m_l}{bw_l}, \quad l \in OR, \tag{3.7}$$

where K is a positive constant, bw_l is the bandwidth requirement of connection l, and $\hat{a}_l$ is the updated value of a_l.

We use the *absorption* variable to keep track of extra slot usage by OR connections. If there are any extra slots left by connections in the set UR, PVQS sorts out connections in the set OR according to the *absorption* variable. Connections with smaller *absorption* values have a higher priority to use extra slots to transmit more cells. After cell transmission, a connection's *absorption* variable is increased and the connection has less chance to use any more slots. In the long run, the differences among *absorption* variables are much smaller than variables themselves.

The complete description of the PVQS scheduling algorithm is as follows:

Step 1: Pre-calculate the ration for each connection.

Step 2: Initialize grant variable g and absorption variable a to zero for each connection and add them to the corresponding lists of $\{g_i\}$ and $\{a_i\}$.

Step 3: Start a new frame.

Step 4: Inspect each connection queue, classify connections into two types of UR

and OR, update the grant list $\{g_j\}$, and calculate the extra slot es as follows.

4.1: For each connection j, if $n_j > r_j, j \in OR$; Otherwise $j \in UR, g_j = g_j + (r_j - n_j)$;

4.2: Move g_j toward the beginning of grant list $\{g_j\}$ so that elements in the list $\{g_j\}$ are in descending order;

4.3: $es = \sum_{i \in UR,OR}(N - r_i) + \sum_{j \in UR}(r_j - n_j)$.

Step 5: Schedule cells out according to the ration variable r for each connection.

Step 6: Find the first OR connection identifier k in the grant list $\{g_k\}$, and perform the following operations.

6.1: $m_k = min(es, n_k - r_k), g_k = g_k - m_k, es = es - m_k$;

6.2: Move g_k toward the end of grant list $\{g_k\}$ so that elements in the list $\{g_k\}$ are in descending order;

6.3: If $(es > 0)$, find the next OR connection identifier until the end of grant list $\{g_k\}$ and repeat the above procedures in the Step 6.

Step 7: Find the first OR connection identifier k in the absorption list $\{a_k\}$, and perform the following operations.

7.1: $m_k = min(es, u_k), a_k = a_k + K * m_k/bw_k, es = es - m_k$;

7.2: Move a_k toward the end of absorption list $\{a_k\}$ so that elements in the list $\{a_k\}$ are are in ascending order;

7.3 If $(es > 0)$, find the next OR connection identifier until the end of absorption list $\{a_k\}$ and repeat the above procedures in the Step 7.

Step 8: Go back to the Step 3.

If we did not normalize extra slot usage by the bandwidth, we would have unused bandwidth evenly allocated to all connections in the set OR. This is often called fair queueing. Fair queueing is a special case which can be easily handled by PVQS. Proportional allocation of unused bandwidth is a distinctive feature which differentiates PVQS from ordinary fair queueing disciplines.

In summary, per-VC queueing scheduler (PVQS) schedules cell transmission on a frame basis. Any unused slots will be *immediately* exploited to serve other connections. PVQS first selects cells under connections' rations. It then allocates any extra slots available to schedule cell transmission for connections that have unused rations. Finally, PVQS checks the *absorption* variables to dispatch more cells than connections' rations. PVQS operates on the cell level to enforce bandwidth requirement on the connection level. In essence, bandwidth is guaranteed for *all* connections. Because of utilization of excess capacity, PVQS can provide a temporarily higher throughput than the bandwidth a connection requested during the setup time. This feature is well suited to bursty data traffic, because it provides a fast draining mechanism when bursts arrive in a VC queue.

3.4 Performance Analysis

In this section, we conduct performance analysis of PVQS. We focus on the following key quality of service (QoS) parameters: delay, delay jitter, throughput, and buffer occupancy distribution. For delay-sensitive traffic, service guarantee is provided for delay,

delay jitter, and throughput. For delay-insensitive traffic, service guarantee is provided only for throughput. Analysis of buffer occupancy distribution is used to calculate cell loss probability for our proposed per-VC queueing ATM switches.

It is assumed that per-VC queueing is implemented in ATM switches. A connection has to specify or negotiate a rate during the connection's setup time and call admission control is executed before a connection is admitted. As outlined in [74], delay-sensitive connections use peak-rate in their bandwidth specification, while delay-insensitive connections use average-rate in their bandwidth specification. To ensure stability, it is assumed that the link capacity is greater than the sum of rates of admitted connections, i.e., $C_L > \sum_i bw_i$. Load conditions of system can be stringent as long as the above condition is not violated.

3.4.1 Delay and delay jitter bounds

In a previous section, we proposed a logical frame structure of N cell slots for a link with capacity C_L. The frame duration is denoted as T. Now we derive the end-to-end delay bound for delay-sensitive connections, assuming PVQS is deployed in all enroute ATM switches, and the same N-slot frame structure and link capacity C_L are used within the network. The end-to-end delay is defined as the time elapsed between the arrival of a cell at the first-hop access switch and the receipt of the cell at the last-hop access switch.

First we obtain the delay bound at the first-hop access switch. We assume that there are k delay-sensitive connections whose peak-rate summation is less than or equal to C_L/N,

i.e.,

$$\sum_{i=1}^{k} bw_i \leq \frac{C_L}{N}, \tag{3.8}$$

where bw_i is the bandwidth for connection i. Because the rate summation of the above k connections is not greater than the bandwidth allocated to one slot in an N-slot frame structure, we can arrange cells from these k connections to share the same slot position in consecutive frames.

There are two components for the delay experienced by a cell arriving at the first-hop access switch: the waiting time to be transmitted and the service time for the transmission of the cell. We first calculate the waiting time encountered by the cell. In the worst case, one cell from each of the k connections arrives at the same time, and they all miss the start of a frame. They have to wait for the start of the next frame. We mark the cell from one connection as "tagged". This "tagged" cell has to wait for the finish of cell transmission from other $(k-1)$ connections, and therefore the "tagged" cell experiences the longest waiting time, which is

$$d_w \leq T + (k-1)T + T = (k+1)T. \tag{3.9}$$

The first term in the above equation represents the upper bound of waiting time for the start of a new frame, the second term is the time elapsed to transmit $(k-1)$ cells in the same slot position, and the last term is the upper bound between the start of frame and the start of transmission of the "tagged" cell.

Now we calculate the second component of the delay at the first-hop access switch, which is the cell transmission time. Let l denote the cell length, the cell transmission time

m is

$$m = \frac{l}{C_L}. \tag{3.10}$$

Combining the above two terms, we can express the delay d_{1a} at the first-hop access switch as

$$d_{1a} = d_w + m \leq (k+1)T + \frac{l}{C_L}. \tag{3.11}$$

Since cells from the above k connections share the same slot position, the superposition of these k connections will be treated as a single connection coming out of the access switch and going into core of the network. Reusing Eqn. (3.9), we have delays introduced at the core switches with a maximum of $2T$ each. If there are h hops between the first-hop access switch and the last-hop access switch, the maximum delay introduced by switches other than the first one will be $h \times 2T$. The delay introduced by all switches is given by

$$d_{sw} = d_{1a} + h \times 2T \leq (k+1)T + \frac{l}{C_L} + h \times 2T. \tag{3.12}$$

Adding the propagation delay $prop$ to the above d_{sw}, we will attain the end-to-end delay d,

$$d = prop + d_{sw} \leq prop + (k+1)T + \frac{l}{C_L} + h \times 2T. \tag{3.13}$$

Based on the above analysis and derivation, we can easily obtain the upper bound and lower bound of the end-to-end delay d:

$$d_{max} \leq prop + (k+1)T + \frac{l}{C_L} + h \times 2T \tag{3.14}$$

and

$$d_{min} \geq prop + \frac{l}{C_L}. \tag{3.15}$$

The delay jitter dj is defined as the difference between d_{max} and d_{min}. The upper bound for delay jitter can be derived from Eqn. (3.14) and Eqn. (3.15) as follows:

$$\begin{aligned} dj &= d_{max} - d_{min} \\ &\leq [prop + (k+1)T + \frac{l}{C_L} + h \times 2T] \\ &\quad -[prop + \frac{l}{C_L}] \\ &\leq (k+1)T + h \times 2T. \end{aligned} \tag{3.16}$$

If the rate of a connection is so large that at least two cells have to be forwarded in a frame, we can decompose the connection into sub-connections, such that we can use the above derivation to obtain similar results.

3.4.2 Throughput

Per-VC queueing scheduler (PVQS) has attractive properties and they can be expressed by the following lemmas and theorem.

Lemma 1: PVQS provides bandwidth guarantee for every connection.

Proof: For an outgoing link with capacity C_L and N (logical) slots in a frame, we allocate

slot ration r_i for connection i according to Eqn. (4.1):

$$r_i = \frac{bw_i}{C_L} \times N, \tag{3.17}$$

where bw_i is the bandwidth requirement of connection i.

The bandwidth allocated by PVQS to connection i is

$$\begin{aligned} bw &= \frac{r_i}{N} \times C_L \\ &= \frac{bw_i}{C_L} \times N \times \frac{1}{N} \times C_L \\ &= bw_i \qquad \textbf{Q.E.D.} \end{aligned}$$

Lemma 2: PVQS is fair in the sense that it allocates any extra bandwidth in proportion to requested rates of OR connections.

Proof: Let I and J represent extra slots consumed by OR connections i and j during the interval $(0, T_l)$. Ignoring the small difference between *absorption* variables a_i and a_j, we have

$$\begin{aligned} a_i &= a_j, \\ K \times \frac{I}{bw_i} &= K \times \frac{J}{bw_j}, \\ \frac{I/T_l}{J/T_l} &= \frac{bw_i}{bw_j}. \qquad \textbf{Q.E.D.} \end{aligned}$$

Theorem: For OR connection i, the extra bandwidth received beyond the requested

bandwidth bw_i is

$$\frac{bw_i}{\sum_i bw_i} \times (C_L - \sum_i bw_i - \sum_j bw_j),$$
$$i \in OR, \quad j \in UR.$$

Proof: From **Lemma 2**, any bandwidth capacity left, which is $C_L - \sum_i bw_i - \sum_j bw_j, \quad i \in OR, \quad j \in UR$, will be distributed in proportion to OR connections' requested rates. For a OR connection i which has requested a bandwidth bw_i, its portion in total requested bandwidth is $bw_i / \sum_i bw_i$, therefore connection i can benefit from the following extra bandwidth beyond bw_i:

$$\frac{bw_i}{\sum_i bw_i} \times (C_L - \sum_i bw_i - \sum_j bw_j),$$
$$i \in OR, \quad j \in UR. \qquad \textbf{Q.E.D.}$$

3.4.3 Buffer occupancy

We use a discrete-time model and assume that link transmission is time-slotted, with each time slot being the transmission time of an ATM cell. Note that the analysis presented here does not depend on a particular scheduling policy as long as it is work-conserving.

The arrival process on the input ports are assumed to be bursty. To represent burstiness of the incoming traffic, we use the so-called ON/OFF model as shown in Figure 3.4. The traffic on each input port is assumed to have two states: an ON state in which a cell arrives in every slot, and an OFF state in which no cell arrives. Each input port alternates

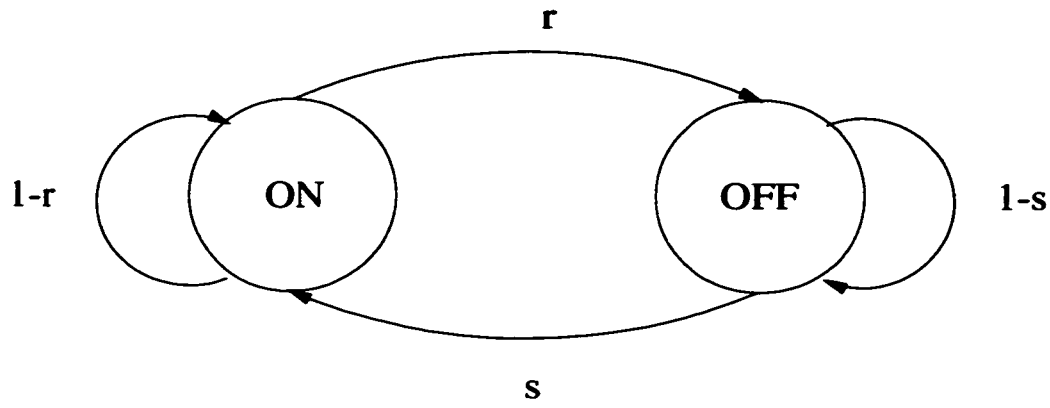

Figure 3.4: ON/OFF model for bursty data traffic.

between ON and OFF states. Let r denote the state transition probability from an ON to an OFF state, and s from an OFF to an ON state. The average cell arrival rate, or the offered load ρ, of each input port is therefore

$$\rho = \frac{1/r}{1/r + 1/s} = \frac{s}{r+s}.$$

In a per-VC queueing switch as shown in Figure 3.2, the Scheduling Module j ($1 \leq j \leq N$) for the output port j selects one of the M_j address lists, uses the cell address (token) referenced by the HEAD register to read out a cell from the central buffer, and sends it to the output port j. Therefore only one cell is sent out to the output port in any slot time by the Scheduling Module. We can imagine that cells referenced by the address tokens in those M_j address lists form a virtual queue for an output port j. Scheduling Module j is just smart enough to pick out any cell in this virtual queue, and sends it out to the output port j. In the next subsection, we shall use the above virtual queue technique to derive

the buffer occupancy distribution and cell loss probability under bursty traffic sources. We assume that each incoming cell has equal probability $1/N$ of being addressed to any given output port as in [27].

Virtual queue length distribution

To derive the virtual queue length distribution for each output port, we need to analyze the cell arrival process from all input ports. Let n, the number of input ports in the ON state, denote the state of the cell arrival process. Let P_{ij} denote the state transition probability of changing from i ON-sources (input ports) to j ON-sources between successive cell slots. Suppose that there are i ON-sources and $N-i$ OFF-sources. Of those i sources in the ON state, we assume that k sources make the transition with probability r to the OFF state in the next slot, and $i-k$ sources stay in the ON state with probability $1-r$. Similarly, of those $N-i$ sources in the OFF state, we assume that l sources make the transition with probability s to the ON state in the next slot, and $N-i-l$ sources stay in the OFF state with probability $1-s$. In the next slot the number of ON sources j will be

$$j = i - k + l. \tag{3.18}$$

The state transition probability from i to j sources in the ON state is

$$P_{ij} = \sum_{k=0}^{i} \binom{i}{k} r^k (1-r)^{i-k} \binom{N-i}{l} s^l (1-s)^{N-i-l}, \tag{3.19}$$

with $l = j - i + k$ from Eqn. (3.18).

Let $\boldsymbol{\pi} = [\pi_0 \ \ \pi_1 \ \ ... \ \ \pi_N]$ denote the steady-state probability of the number of sources in the ON state, and $\mathbf{P} = [P_{ij}]$ the state transition matrix, we can obtain $\boldsymbol{\pi}$ by solving the following balance equation:

$$\boldsymbol{\pi} = \boldsymbol{\pi P}. \tag{3.20}$$

We fix our attention on a particular virtual output queue, and note that each cell has equal probability $1/N$ of being addressed to any given output port. Let a be the number of arrivals during a given cell slot, then we have

$$a_i = Pr[a = i] = \sum_{n=0}^{N} \binom{n}{i} (1/N)^i (1 - 1/N)^{n-i} \pi_n, \quad i = 0, 1, \ldots, n. \tag{3.21}$$

After some manipulations, the probability generating function (PGF) of a can be obtained as:

$$A(z) = \Pi(1 - \frac{1}{N} + \frac{z}{N}), \tag{3.22}$$

where $\Pi(z) = \sum_{n=0}^{N} \pi_n z^n$, the PGF of $\boldsymbol{\pi}$.

Letting q_m demote the number of cells at the end of mth time slot, and a_m be the number of cell arrivals during the mth time slot, we have

$$q_m = \max(0, \ \ q_{m-1} + a_m - 1). \tag{3.23}$$

Using a standard technique in discrete-time queueing analysis as in [39], we can derive

the PGF for the steady-state queue size as:

$$Q(z) = \frac{(1-\rho)(1-z)}{A(z)-z}. \tag{3.24}$$

Differentiating Eqn. (3.24) with respect to z and taking the limit as $z \to 1$, we obtain the mean $\overline{q}$ and variance σ_q^2 of steady-state queue size.

Distribution of buffer occupancy

The buffer size requirement of the proposed per-VC queueing architecture is the same as the complete sharing switch presented in [27], because two switches operate in the same way as far as buffering is concerned. However, we use a different approach to derive a *closed-form* formula to approximate the buffer occupancy distribution and the relationship between the cell loss probability and the buffer size.

The buffer occupancy b_m at the end of the mth slot is the summation of all output queue lengths $q_{i,m}, i = 1, \ldots, N$, i.e.,

$$b_m = \sum_{i=1}^{N} q_{i,m}. \tag{3.25}$$

The buffer occupancy b_m will reach a steady-state distribution as m approaches infinity. Let $B_m(z)$ represent the probability generating function of b_m, we have

$$\lim_{m \to \infty} B_m(z) = B(z), \tag{3.26}$$

where $B(z)$ is the probability generating function of buffer occupancy b.

If we assume the same ON-OFF bursty traffic on all input ports and an infinite buffer size, the queue sizes $q_i, i = 1, \ldots, N$ become independent identically distributed (iid) random variables. As N increases, the steady-state buffer occupancy b approaches Gaussian distribution,

$$\begin{aligned} Pr[b < B] &= Pr\left[\frac{b - N\overline{q}}{\sqrt{N}\sigma_q} < \frac{B - N\overline{q}}{\sqrt{N}\sigma_q}\right] \\ &= \int_{-\infty}^{\frac{B-N\overline{q}}{\sqrt{N}\sigma_q}} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \end{aligned} \tag{3.27}$$

where $\overline{q}$ and σ_q^2 are the mean and variance of the steady-state queue size derived from Eqn. (3.24).

With the assumption of an infinite buffer size, we then approximate the cell loss probability by $Pr[b > B_{real}]$, where B_{real} denotes the real buffer size. Note that the above probability gives an upper bound of cell loss probability. From the above buffer occupancy distribution, we can easily obtain the minimum buffer size B_{min} required to achieve a desired cell loss probability, say $10^{-k}, k = 1, 2, \ldots$. This is shown in the following.

Let u_k be the value which satisfies

$$\int_{u_k}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = 10^{-k}, \quad k = 1, 2, \ldots. \tag{3.28}$$

Note that the values of $u_k, k = 1, 2, \ldots$ are well tabulated [33].

Using Eqn. (3.27), we can calculate the cell loss probability and set it to 10^{-k},

$$\begin{aligned} Pr[\text{cell loss}] &= Pr[b > B_{min}] \\ &= Pr\left[\frac{b - N\overline{q}}{\sqrt{N}\sigma_q} > \frac{B_{min} - N\overline{q}}{\sqrt{N}\sigma_q}\right] \\ &= \int_{\frac{B_{min}-N\overline{q}}{\sqrt{N}\sigma_q}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \\ &= 10^{-k}. \end{aligned} \tag{3.29}$$

Comparing Eqns. (3.28) and (3.29), we obtain the minimum buffer size B_{min} to achieve a desired cell loss probability 10^{-k},

$$B_{min} = u_k\sqrt{N}\sigma_q + N\overline{q}, \tag{3.30}$$

where $\overline{q}$ and σ_q^2 are the mean and variance of the steady-state queue size.

3.5 Performance Evaluation

Based on the previous analysis, we evaluate the performance of PVQS through numerical examples in this section.

The first example illustrates how to calculate the delay and delay jitter bounds for transporting a 6 Mb/s connection over a 622 Mb/s (OC-12) ATM network across continental USA. The number of hops is assumed to be 5, and the propagation delay is assumed to be 30 ms.

link capacity (OC-12)	$C_L = 622$ Mb/s
number of hops	$h = 5$
prop. delay across USA	$prop = 30$ ms
# of slots in a frame	$N = 10$
bandwidth per slot	62.2 Mb/s
maximum k for 6 Mb/s	$k = 10$
cell length (bits)	$l = 424$
N-slot frame duration	$T = 6.8\ \mu$ s
delay by switches	0.2 ms
end-to-end delay	30.2 ms
delay jitter bound	0.2 ms

Table 3.1: Calculation of end-to-end delay and delay jitter bounds.

From Tables 3.1 and 3.2, we notice that the end-to-end delay is dominated by the propagation delay (30ms) while delay introduced by enroute switches is just a small fraction of the total delay in both $N = 10$ and $N = 100$ cases. The end-to-end delay is not sensitive to the choice of N, the number of slots in a frame. The above table clearly establishes a significant advantage of introducing an N-slot frame structure to ensure a low maximum delay in comparison with the end-to-end propagation delay.

The second example illustrates the proportional allocation behavior of PVQS. The unit for bandwidth is Mb/s in Tables 3.3 and 3.4.

Assume that the link capacity C_L is 622 Mb/s (OC-12). There are four connections over this link: connection #1 uses the peak rate of 82 Mb/s as its bandwidth requirement. Three other connections request rates of 80 Mb/s, 160 Mb/s, and 240 Mb/s respectively. If connection #1 does reach its peak rate, the initial bandwidth left will be $622 - 82 - 80 - 160 - 240 = 60$ Mb/s, and it will be distributed to the connections #2, #3, and #4

link capacity (OC-12)	$C_L = 622$ Mb/s
number of hops	$h = 5$
prop. delay across USA	$prop = 30$ ms
# of slots in a frame	$N = 100$
bandwidth per slot	6.22 Mb/s
maximum k for 6 Mb/s	$k = 1$
cell length (bits)	$l = 424$
N-slot frame duration	$T = 68\ \mu$ s
delay by switches	0.8 ms
end-to-end delay	30.8 ms
delay jitter bound	0.8 ms

Table 3.2: Calculation of end-to-end delay and delay jitter bounds.

link capacity (OC-12)	$C_L = 622$
peak rate for conn #1	82
req. *bw* for conn #2, #3, #4	80, 160, 240
actual rate for conn #1	82
initial bandwidth left	60
bw for conn #2	090
bw for conn #3	180
bw for conn #4	270

Table 3.3: Proportional bandwidth allocation.

according to the ratio of 1:2:3. The extra bandwidth received by these connections will be 10 Mb/s, 20 Mb/s, and 30 Mb/s. The total bandwidth allocated to the three connections will be 90 Mb/s, 180 Mb/s, and 270 Mb/s. If the actual bandwidth consumption of the connection #1 is 52 Mb/s (below peak rate), the initial bandwidth left will be increased to 90 Mb/s, and it will be distributed to the three connections according to the ratio of 1:2:3. The extra bandwidth received by the connections #2, #3, and #4 will be 15 Mb/s, 30 Mb/s, and 45 Mb/s. The total bandwidth allocated to the three connections will be 95

link capacity (OC-12)	$C_L = 622$
peak rate for conn #1	82
req. *bw* for conn #2, #3, #4	80, 160, 240
actual rate for conn #1	52
initial bandwidth left	90
bw for conn #2	095
bw for conn #3	190
bw for conn #4	285

Table 3.4: Proportional bandwidth allocation.

Mb/s, 190 Mb/s, and 285 Mb/s. The two scenarios are summarized in Tables 3.3 and 3.4.

To verify the soundness of our analysis, we conduct simulations and results obtained are compare against the analytical ones. We choose to re-use the same network setup and parameter configuration as described below as in [82].

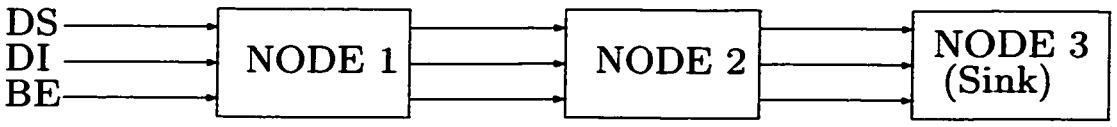

Figure 3.5: A three-node ATM network.

Figure 3.5 represents the network model used in the simulation study. The network is composed of three nodes in a linear topology connected by high-speed OC-192 links. Each node is modeled with connections from the upstream node if applicable. The third node acts as sinks for all connections.

The traffic model used in this simulation study is the following. For delay-sensitive (DS) connections a Bernoulli generation model is employed. In each slot, cell arrival probability is determined by the Bernoulli distribution. For delay-insensitive (DI) and best-effort (BE)

Parameter	Value
Number of repetitions	100000 frames minimum
link capacity	23600000 cells/s
Fraction of DS cells	0.25
Fraction of DI cells	0.50
Fraction of BE cells	0.25
K (absorption coefficient)	20
Link delay (node1 to node 2)	32 cell times
Link delay (node2 to node 3)	32 cell times
Node buffer	236000 cells

Table 3.5: Parameters used in the simulation

connections, an ON-OFF bursty source is employed to model cell arrival. This model is described as being composed of two states: an ON state in which cells arrive in consecutive slots and an OFF state in which no cell arrives.

In our simulations, the proportion of DS:DI:BE connections is 1:2:1. All the simulations are performed on a Sun SPARC 5 workstation. Simulations run continuously until the 90% confidence interval for the mean narrows to a desired width which is pre-defined as 10% of the mean. A typical run time is about 10 hours. Table 3.5 summarizes some of the key parameters that are used in the simulations.

3.5.1 Delay and delay jitter

The simulation results of delay distribution for two delay-sensitive connections are demonstrated in Figures 3.6 and 3.7. In Figure 3.6, the mean delay is 98.3 and the variance is 26.7. In Figure 3.7, the mean delay is 99.6 and the variance is 27.2. Other delay-sensitive connections exhibit similar patterns and for the reason of brevity they are not shown. To

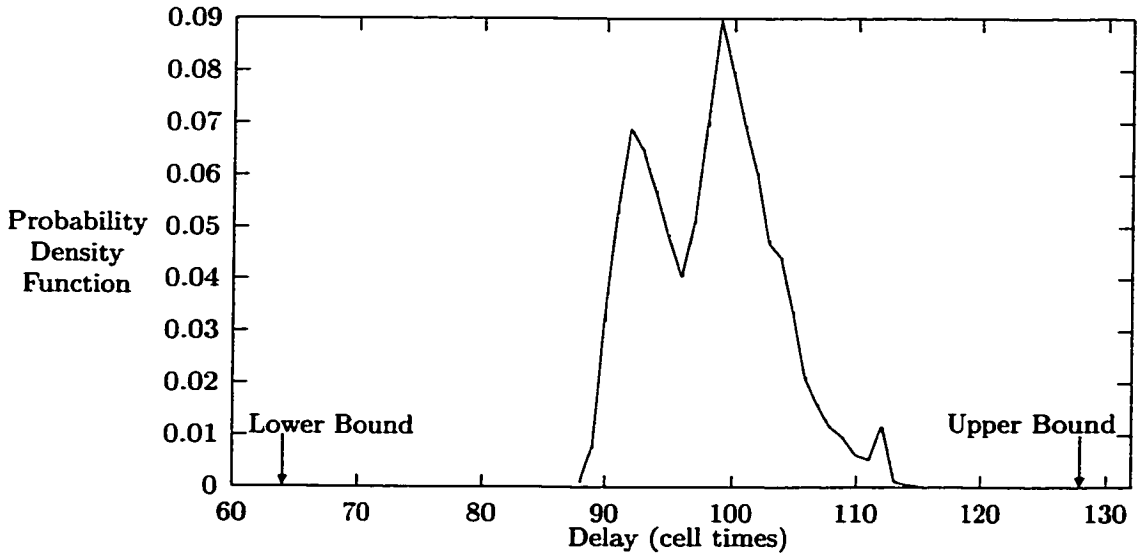

Figure 3.6: Delay distribution for DS connections.

compare with analytical results derived from previous sections, both delay lower bound and delay upper bound are marked in the Figures 3.6 and 3.7 as well. It is observed that all delay-sensitive cells arrive at their destination (the third node) between the lower and upper bounds. Since all delay-sensitive cells arrive within the derived delay bounds, it is straightforward to show that delay jitter falls into the jitter bound, due to the fact that delay jitter is defined as the difference between the upper and lower delay bounds.

3.5.2 Throughput

Table 3.6 presents the simulation results of throughput for delay-sensitive and delay-insensitive connections. All throughput values are very close to 100% as expected. At the end of a simulation run, there is a tiny percentage of DS and DI cells staying in the

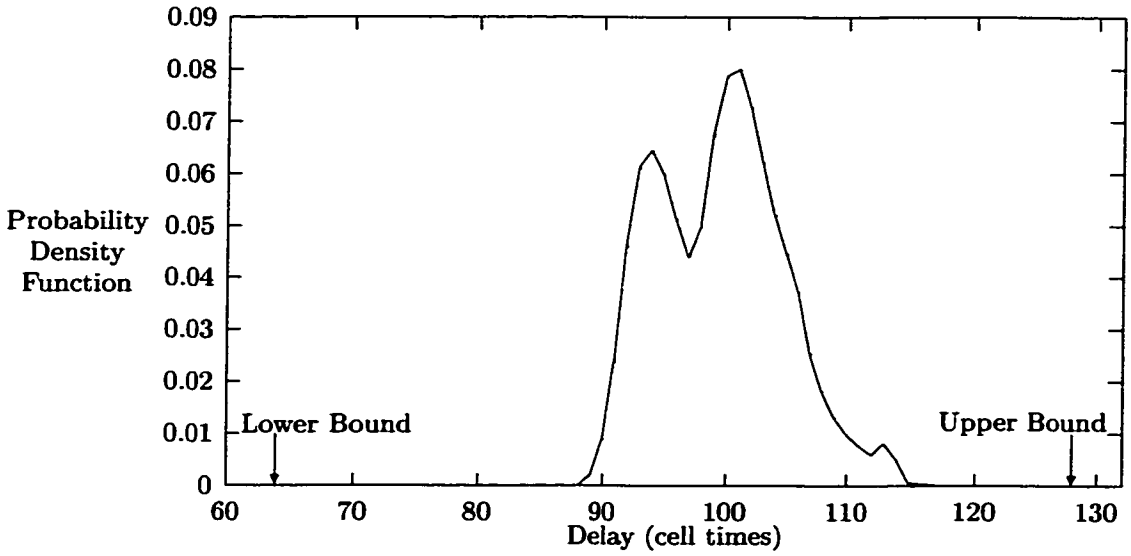

Figure 3.7: Delay distribution for DS connections.

node buffers. These cells account for the discrepancy between the simulation results in Table 3.6 and expected 100%.

Connection	Throughput
DS #1	99.93%
DS #2	99.97%
DI #1	99.97%
DI #2	99.99%
DI #3	99.95%
DI #4	99.96%

Table 3.6: Throughput for DS and DI connections

3.5.3 Buffer occupancy

In this subsection, we use both numerical and simulation results to understand the buffer performance of proposed per-VC queueing ATM switches under bursty traffic. We select parameters s and r of the ON/OFF model such that they represent both bursty traffic source and heavy offered load. The state transition probability r from ON to OFF state is chosen to be 0.1, and the state transition probability s from OFF to ON state is chosen to be 0.5. The average burst length $1/r$ is 10 cells, and the average idle period $1/s$ is 2 cells. The offered load $\rho = (1/r)/(1/r + 1/s)$ is 0.83, which is high. We can see that the above traffic is quite bursty and heavy loaded.

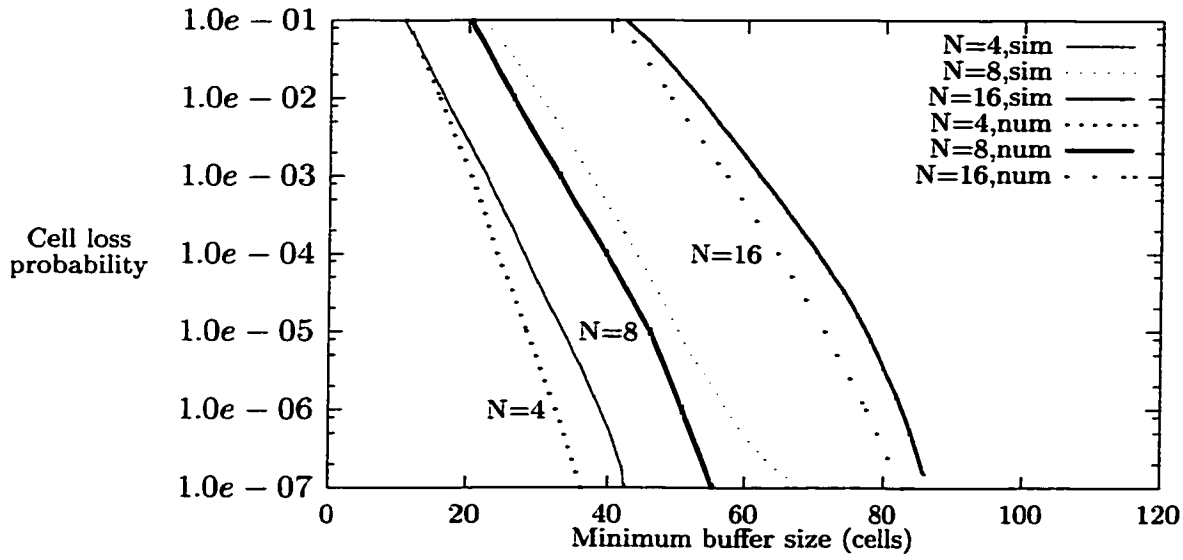

Figure 3.8: Minimum buffer size to achieve a desired cell loss probability.

Figure 3.8 shows the performance of cell loss probability under various buffer size. As a validation of previous analysis modeling, simulation results for switch size $N = 4, 8, 16$ are used to compare with numerical results which can be obtained from analysis. It can be

seen that analytical results derived in previous sections are a good approximation to the simulation ones. As expected, the buffer performance under bursty traffic is worse when compared with uniform traffic [76].

3.6 Complexity Analysis

For per-cell based scheduling algorithms, they require assigning a virtual time-stamp to each incoming cell and then sorting all buffered cells according to the values of their time-stamps. Therefore, the complexity of per-cell based scheduling algorithms is in the order of

$$O(log(b) \times \alpha \times t), \tag{3.31}$$

where b is the buffer size, α the aggregated cell arrival rate, and t the time interval.

For our proposed per-connection based approach, it requires the calculation of grant and absorption variables. The computation is not done on a per-cell basis, but rather on an N-slot frame basis. After the calculation is done, it is necessary to sort connections according to the values of their grant and absorption variables. Therefore, the complexity of per-cell based scheduling algorithms is in the order of

$$O(log(n) \times \alpha \times t), \tag{3.32}$$

where n is the number of existing connections, α the aggregated cell arrival rate, and t the time interval. If we assume that $log(b)$ is in the same order of $log(n)$, the complexity of

per-cell based approaches is about $O(N)$ times of per-connection based method, where N is the number of slots in a frame.

Per-connection based scheduling has drawbacks as well. Since sorting is performed on existing connections, the number of admitted connections is a limiting factor which adds constraints on the scalability of per-connection oriented approach. In other words, it is impossible to handle a large number (e.g. a million) of connections by our proposed scheduling algorithm. For per-cell based approaches, the limiting factor is the requirement to finish the processing of an incoming cell before the arrival of next ATM cell. This adds constraints on the speed of line interface where ATM cells arrive from. Put it in another way, the duration of a slot, which corresponds to an ATM cell arrival, has to be large enough to allow processing to be finished before the arrival of the next slot. Therefore, the line interface speed can not go beyond a certain value.

3.7 Concluding Remarks

In this chapter, we proposed a connection-oriented ATM switching architecture which provided per-VC queueing capability. This fundamental change of switching architecture from port-oriented design to connection-oriented design led to a number of developments, one of which is a proposal of per-VC queueing scheduler (PVQS). PVQS is a feasible scheduling algorithm for per-VC queueing ATM switches. The feasibility of PVQS could lead to a practical VLSI implementation of PVQS in ATM switching chips.

Chapter 4

Switching and Scheduling in IP over ATM Networks

As discussed in the introduction, the next-generation IP networks have to introduce the concept of QoS and provide levels of service that correspond to different classes of traffic. Real-time interactive voice and video applications impose a stringent delay requirement. Data applications demand throughput guarantee. Routers, the basic building blocks in any IP networks, have to identify different traffic types and handle each of them adequately. In this chapter, we present our insight to the issue of providing QoS guarantee in IP networks. Switching and scheduling are still the key ingredients in any IP QoS schemes. Based on ATM's per-VC queueing architecture, we introduce integrated frame and cell switching. On the scheduling front, we extend Per-VC Queueing Scheduler (PVQS) and present per-flow scheduling (PFS).

4.1 Per-flow Queueing in Access Routers

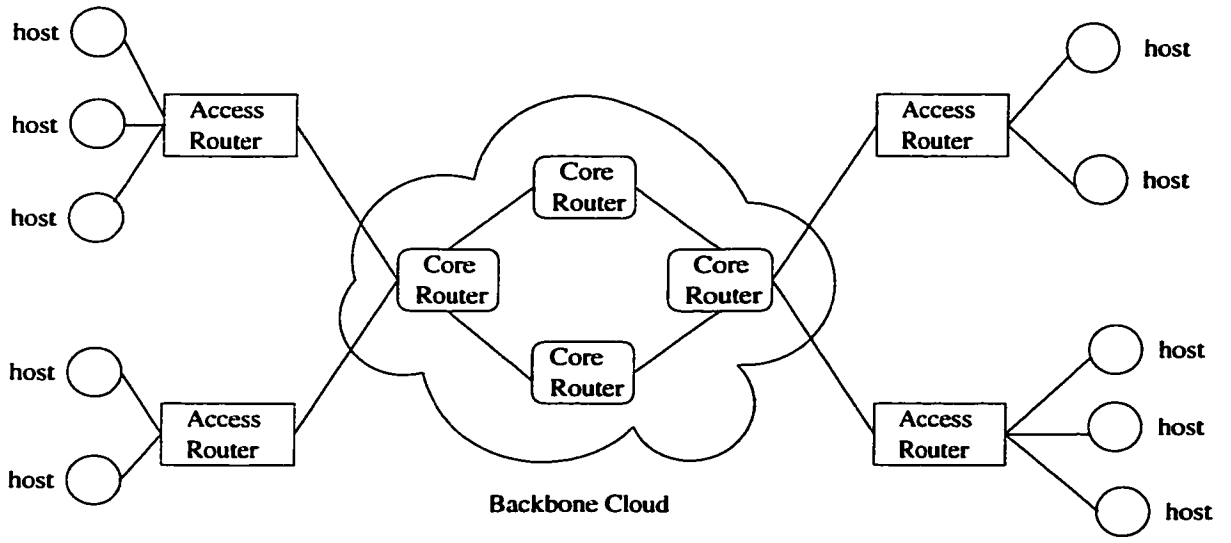

Figure 4.1: A general IP network with hosts, access routers, and core routers.

Figure 4.1 illustrates a general topology for an IP network: hosts are attached to access routers, and access routers are connected to a backbone cloud formed by core routers.

Access routers play the role of multiplexing IP traffic from multiple hosts onto the backbone cloud of the network. Under the paradigm of Multi-Protocol Label Switching (MPLS) [8], access routers handle IP traffic flows utilizing a layer-2 switching mechanism. Since we choose ATM as the layer-2 switching technology, IP route information will be mapped into ATM VPI/VCI labels. In this section, we assign a separate VC for each IP flow and use per-flow queueing [75] as the switching architecture in access routers.

Per-flow queueing is a natural evolution of its ATM counterpart of per-VC queueing [77]. Instead of providing a separate queue for each VC in per-VC queueing, per-flow queueing supplies a separate queue for each IP flow and maps an IP flow to an ATM VC if

MPLS is used. After a packet from an IP flow is received by the first-hop access router, the packet is stored directly in a queue allocated for the IP flow *without* segmentation. The packet is then read out from the queue, 48 bytes each time, as payloads for multiple ATM cells (padding bytes may be needed in the last cell). As these cells traverse in the core of the network, cell headers steer cells into a dedicated queue associated with the IP flow. At the last-hop access router, each queue only collects cells which belong to the same flow, therefore packets are *automatically* reassembled in the queue. Per-flow queueing creates a virtual pipe for each IP flow, and facilitates routers to provide high-speed delivery and QoS guarantee.

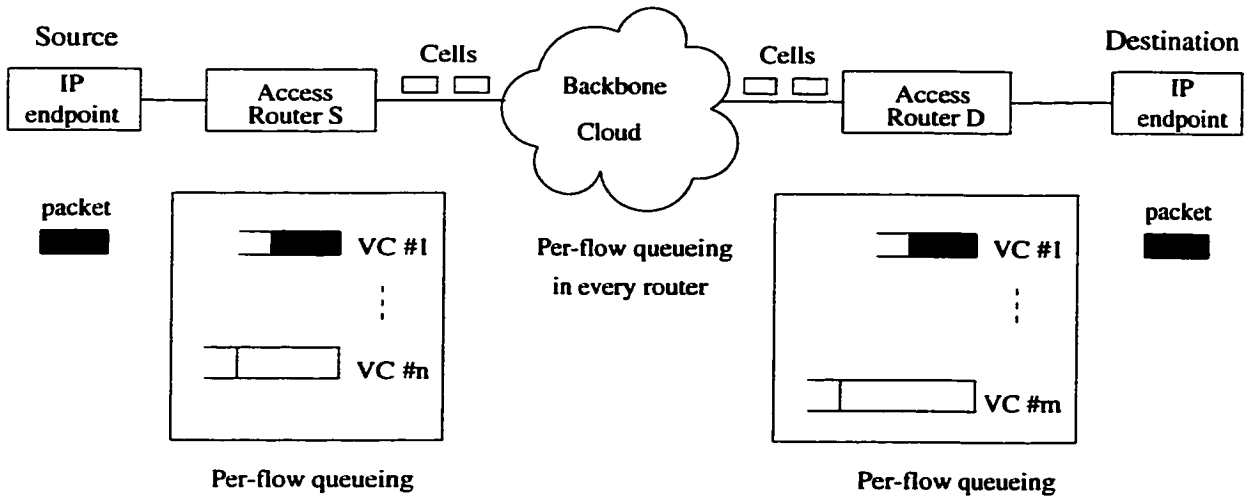

Figure 4.2: A per-flow queueing network.

In this section, we present an implementation of access routers based on the per-flow queueing architecture. Figure 4.2 depicts a general per-flow queueing network. The access routers are used to route IP packets from the source endpoint to the backbone cloud and to the destination endpoint. We assume that routing tables have been set up in routers.

For the clarity of presentation, we temporarily assume that there is no VC merging done by core routers. In the next section, we will remove the assumption and explain why VC merging is necessary.

At first, we show how to route the first IP packet of a new traffic flow. When the first-hop access router (Access Router S) receives such an IP packet, it stores the packet in a queue and assigns an *unused* incoming VC to the queue. The router then looks up its routing table which includes MPLS information and chooses an outgoing link. Subsequently, it selects a VC on the outgoing link and forwards the packet, in a format of multiple ATM cells, to the downstream router (a core router or another access router). The VC selected on the outgoing link acts as a MPLS label, signaling to the downstream router where the destination is for the new flow. After sending out the first IP packet to the downstream router, the upstream router needs to update its connection table by adding a new entry that contains the incoming link ID of the new IP traffic flow, source and destination IP addresses, incoming VC it assigned, outgoing link, outgoing VC it chose, etc., just as in conventional ATM cell switching. When the downstream router receives the multi-cells of the IP packet and recognizes the MPLS label in terms of VPI/VCI in the cell headers, it looks up its MPLS connection table, attaches another MPLS label (VPI/VCI) to the cells, and switches them to the appropriate output port.

The above switching procedure is repeated in the rest of source-to-destination route until the first IP packet of a new flow reaches the last-hop router (Access Router D), which is connected directly to the destination IP endpoint. The last-hop router does not need to send the IP packet in cell format. It just reads out the IP packet from the VC queue

(recall that the packet is automatically reassembled in the VC queue), strips off padding bytes if any, and ships the packet in its native format to the destination. We may use a special outgoing VC, indicating that some special treatments are required to deliver the IP packet in its last hop. These special treatments include stripping off padding bytes, and delivering the packet in its native IP format directly, etc.

In summary, we route the first IP packet of a new flow *only* at the first-hop access router and switch this packet in other routers. When the first IP packet reaches its destination, a connection path is set up at the same time. We then switch all the following packets of the same flow. We emphasize that we use the per-flow queueing technology throughout the routing process, even in the phase of routing the first IP packet.

To distinguish short-lived flows from long-lived ones and to minimize MPLS signaling traffic among routers, we propose that a timer be used for monitoring the connection status in access routers. When the first IP packet is routed at the first-hop access router, a timer is started. This timer is restarted whenever a subsequent IP packet of the same flow is switched. If the timer times out, VC used by the connection on the incoming link is marked as "unused". It will be recycled for new connections later on.

Access routers in Figure 4.2 are similar in spirit to Ipsilon's IP switching [50]. However, our scheme is significantly different from Ipsilon's proposal in that we use *per-flow queueing* as the switching architecture, pick a VC from the *upstream* node, switch *all* IP packets including the first one (the first IP packet of a new flow is *only* routed at the first-hop access router), and make any segmentation and reassembly (SAR) operations *obsolete* inside access routers [75].

4.2 VC Merging for Core Routers

In the previous section, we temporarily assumed that per-flow queueing was also implemented in core routers and there was no VC merging. In this section, we remove the above assumptions and explain why VC merging is necessary.

As mentioned before, ATM is chosen as the layer-2 switching mechanism in MPLS and route information is mapped into ATM VPI/VCI labels. If core routers implement per-flow queueing only, each source-destination pair is mapped to a unique VC value at each core router. The drawback is that it is not scalable because n sources and n destinations would require n^2 labels. This is why per-flow queueing without label merging is only suitable for access routers. To support the fundamental requirement of scalability, core routers have to perform some type of label merging at layer-2. In the context of ATM, VPI/VCI are MPLS labels. There are two basic methods for label merging. The first one is called VP-merging, which maps each destination to a unique VP value. The VC value is used to identify the sender so that packets can be re-assembled correctly at the receiver. However, VP-merging is not practical since the VP space is limited to only 4096 entries at the network-to-network interface (NNI). The second method is called VC-merging, which maps incoming VC labels for the same destination to the same outgoing VC label. This method is scalable and does not have the space constraint problem of VP-merging. However, VC-merging suffers another type of constraint. With VC-merging, cells for the same destination are indistinguishable at the output of a switch. Therefore cells belonging to different packets for the same destination can not be interleaved, or else the receiver will not be able to

reassemble packets. Figure 4.3 illustrates the key difference between VC-merging and non-VC merging [65]. Note that the boundary between two adjacent packets are identified by the End-of-Packet (EOP) indication used by AAL5.

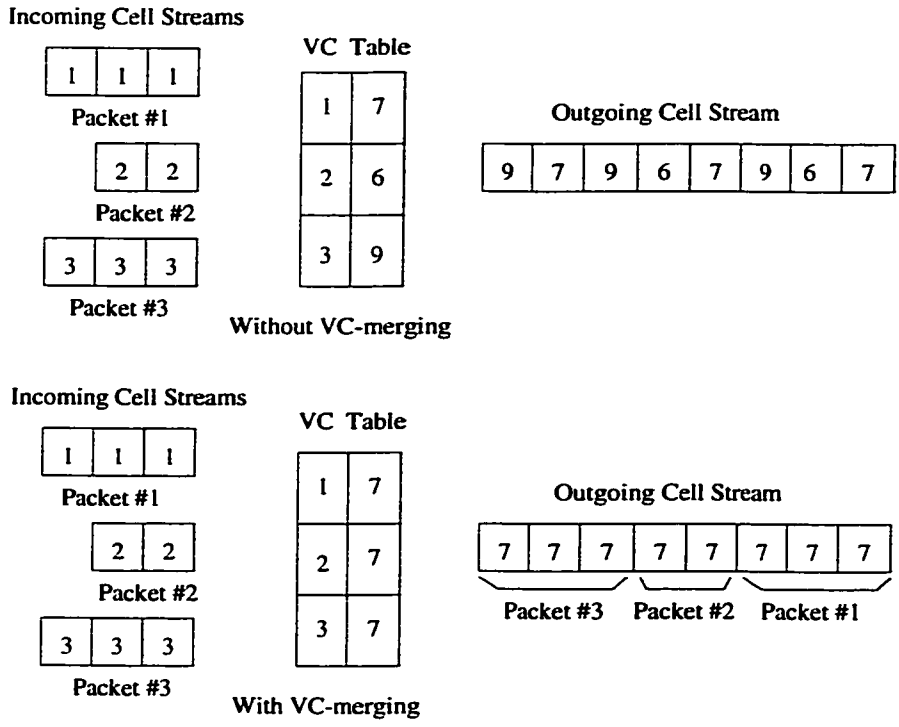

Figure 4.3: Comparison of non-VC merging and VC merging at an output port.

To establish the feasibility that core routers can be implemented by ATM switches, we have to demonstrate that an ATM switch is capable of merging a group of VCs at the output. A prominent requirement of a VC-merge-capable ATM switch is that cells belonging to different packets should not interleave if they are going to be merged to the same outgoing VC. Therefore, each incoming cell for a given packet on an incoming VC needs to be stored in a separate buffer until the last cell of the packet arrives. From our previous experience in per-VC queueing architecture [77], we immediately recognize that

the requirement of not interleaving cells from different incoming VCs can be *perfectly* met by the per-VC queueing which provides a separate queue for each incoming VC!

The only difference between a per-VC queueing cell switch and a VC-merge capable switch is when scheduling starts. In a conventional per-VC queueing cell switch, there is no such concept as packet, and scheduling starts whenever there is a cell available in one of VC queues. In a VC-merge-capable switch however, scheduling is deferred until the last cell for a given packet arrives. Afterwards, all cells belonging to the same packet are scheduled in an atomic manner for transmission to the next hop. Recognizing the only difference between the two types of switches, we can make a simple modification to a per-VC queueing cell switch [77] and convert it into a VC-merge-capable switch. The only modification required for this conversion is to add an End-of-Packet (EOP) detection circuitry to the address list which interrupts a Packet-Ready signal to the Scheduling Module as shown in Figure 4.4. In other words, an arrival of the last cell of a packet triggers the EOP detection circuitry, which in turn asserts the Packet-Ready signal to inform the Scheduling Module that a packet is ready for transmission. After all the cells pertaining to the packet have been sent out, EOP detection circuitry de-asserts the Packet-Ready signal and waits for the completion of the next packet. If there are more than one packet in a queue, the Packet-Ready signal will still be asserted to inform the Scheduling Module that at least one packet is ready. The architecture of per-VC queueing with EOP detection is shown in Figure 4.4. To maintain clarity of our presentation, the actual cell buffer is omitted in Figure 4.4.

It is worthy to note that backward compatibility can be maintained for an EOP-

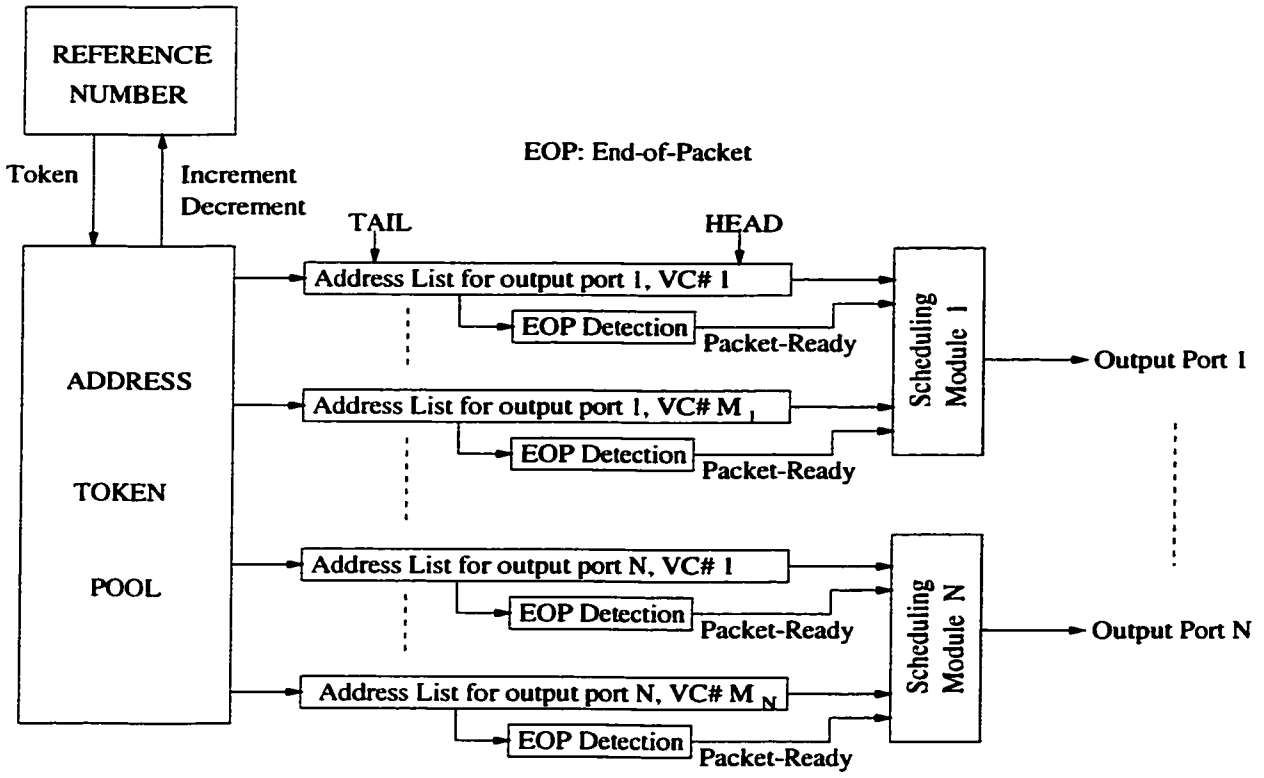

Figure 4.4: Per-VC queueing with EOP detection to support VC-merging.

equipped VC-merge capable switch to support cell switching. If we consider the special case in which a packet consists of only a single cell, we immediately recognize that in order to support cell switching, we can configure EOP detection circuitry so that it triggers on the arrival of a single cell. In this fashion, we can configure a VC-merge capable switch to support *both* VC merging and pure cell switching.

4.3 Scheduling

In this subsection, we extend our previous work in per-VC queueing scheduler [81] and propose a per-flow scheduler (PFS), which resides in every output port of a router. PFS has two important attributes. First, it divides outgoing link capacity among competing IP flows on a per-flow basis and provides a *guaranteed* rate to every flow. Secondly, it proportionally allocates any unused bandwidth.

As described before, we mark each packet within an IP flow using an ATM VPI/VCI label in the access routers, and merge labeled IP flows in the core routers. There is a one-to-one mapping between an IP flow and an ATM VC in the access routers. Multiple flows or ATM VCs from access routers are aggregated into a single flow or VC in the core routers. Therefore one flow from an outgoing port of a core router can be considered as superposition of multiple flows from access routers. PFS deals with flow scheduling in every outgoing port of both access and core routers.

PFS selects packets for transmission on a frame basis (logically) with N packet slots in a frame. It will be shown later that an introduction of N-slot frame structure guarantees not

only a bandwidth, but also a low maximum delay for delay-sensitive flows in comparison with the end-to-end propagation delay.

If the link capacity is C_L and flow i requires a bandwidth of bw_i, the slot ration r_i for flow i in a frame is

$$r_i = \frac{bw_i}{C_L} \times N, \quad i \in \{DS, DI\}, \tag{4.1}$$

where DS designates the set of delay-sensitive flows and DI represents the set of delay-insensitive flows.

At the beginning of a frame, the scheduler for an outgoing link inspects all VC queues associated with the link, and schedules transmission of packets in each VC queue up to the flow's ration. In the case of VC merging in the core routers, a group of VC queues are inspected. The delay-sensitive VCs have a higher priority than delay-insensitive VCs. Recall that we classify traffic into three types: delay-sensitive, delay-insensitive, and best effort. For the clarity of presentation, we assume peak-rate allocation for delay-sensitive flows, and average bandwidth allocation for delay-insensitive flows. Unused bandwidth left by delay-sensitive flows may be allocated to best-effort traffic. As mentioned before, best-effort traffic is ignored in the rest of this thesis in order to maintain clarity of our presentation.

Since we use peak-rate allocation for delay-sensitive flows, packets in delay-sensitive VC queues never exceed their rations. Let n_j be the number of packets in the VC queue for flow j at the beginning of a frame, r_j be the slot ration for flow j, DS represent the

set of delay-sensitive flows, we have

$$n_j \leq r_j, \quad j \in DS. \tag{4.2}$$

For delay-insensitive flows, the numbers of packets at the beginning of a frame can be larger than their rations. We divide the set of delay-insensitive flows DI into two subsets: DIU and DIO. DIU designates the set of delay-insensitive flows for which the numbers of packets at the beginning of a frame are under their rations, i.e.,

$$DIU = \{\text{delay-insensitive flow } k,\ n_k \leq r_k\}. \tag{4.3}$$

DIO represents the set of delay-insensitive flows for which the numbers of packets at the beginning of a frame are over their rations, i.e.,

$$DIO = \{\text{delay-insensitive flow } l,\ n_l > r_l\}. \tag{4.4}$$

From the scheduling point of view, some slots may not be assigned to any flows, and not all slots allocated to flows in DS and DIU are utilized. We can calculate extra slots es as follows:

$$es = (N - \sum_i r_i) + \sum_{j_i} (r_{j_i} - n_{j_i}) + \sum_{k_i} (r_{k_i} - n_{k_i}), \quad i \in \{DS, DI\},\ j_i \in DS,\ k_i \in DIU. \tag{4.5}$$

To achieve multiplexing gain among flows, we need to re-allocate these extra slots left by

flows in the sets DS and DIU, and use them to dispatch packets which belong to flows in the set DIO. In the mean time, flows in DIU should be credited for "freeing up" slots which allows the scheduler to transmit packets from DIO flows. DIU flows should be compensated by being allowed to transmit more packets than rations later on. Data traffic are bursty. When a burst arrives, a flow may experience the transition from the set DIU to DIO, and it is fair for the flow to claim previously unused rations to transmit more packets and get the burst quickly out of the VC queue. In other words, the scheduler should schedule more packet transmissions for flows which have not used up their rations. This should be viewed as part of the scheduler's commitment of providing average bandwidth to every delay-insensitive flow. For this purpose, we introduce a *grant* variable, g_i for flow i, to keep track of unused ration on a per-flow basis. The value of g_i is non-negative. It is initialized as 0 (zero). It is incremented when fewer packets than flow i's ration are scheduled at the beginning of a frame, i.e., the updated value $\hat{g}_k$ is

$$\hat{g}_k = g_k + (r_k - n_k), \quad n_k < r_k, \tag{4.6}$$

and it is decremented when more packets (up to g_k) than flow i's ration are scheduled during a frame,

$$\hat{g}_k = g_k - m_k, \quad m_k < g_k, \tag{4.7}$$

where m_k is the number of packets transmitted over flow k's ration.

The use of *grant* variable is a very desirable feature for bursty traffic, because it boosts instantaneous throughput when a burst of packets arrives. Delay-insensitive traffic tend

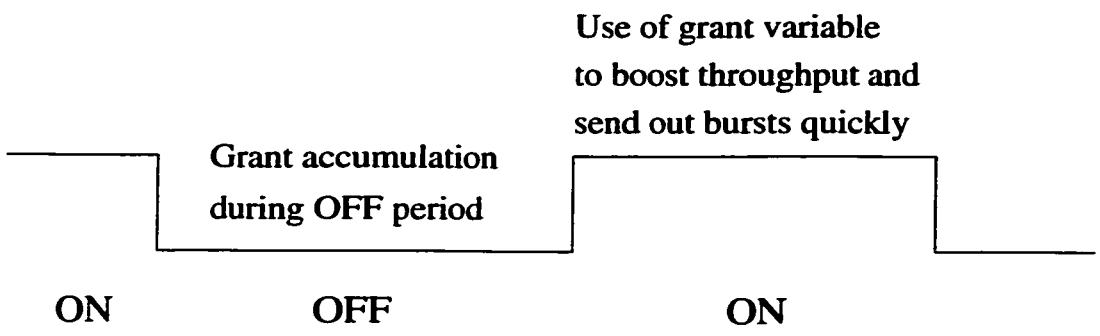

Figure 4.5: Use of *grant* variable to boost instantaneous throughput.

to be bursty and the so-called ON/OFF model is often used to represent bursty traffic. During OFF periods, there is no packet arrival and *grant* is accumulated to record unused rations. During ON periods when bursts of packets arrive, the *grant* variable authorizes the scheduler to send out more packets than what the ration allows, effectively boosting instantaneous throughput to drain bursts out of the queue much more quickly. This scenario is illustrated in Figure 4.5.

Per-flow scheduler (PFS) calculates extra slots available in each frame, and selects DIO flows which have *grants* available for transmitting more packets than their rations. The selection is based on the *grant* variable. The larger value of *grant*, the higher priority a flow has. If *grants* are used up for all flows but extra slots are still not exhausted, the scheduler will select flows in the set DIO for packet transmission. Based on the "pay per bandwidth" billing policy that end users pay more for higher bandwidth, the scheduler will distribute extra slots based on the bandwidth requirement. The higher bandwidth a flow requires, the higher priority the flow has. Here we need to introduce another *absorption* variable to keep track of extra bandwidth consumption on a per-flow basis. The *absorption*

variable, a_i for flow i, is non-negative. It starts at 0 (zero), and it is incremented when a flow takes advantage of extra slots available in a frame to transmit more packets. The smaller value of *absorption*, the higher priority a flow has in consuming any extra slots. To maintain fairness among flows which have required different bandwidths and to reflect tariff imposed on different rates, any extra bandwidth left by flows in the sets DS and DIU should be distributed in proportion to bandwidth requirements for flows in the set DIO. In other words, extra slot usage needs to be normalized by the bandwidth requirement. If m_l extra slots are used by flow l, its absorption variable a_l should be updated as follows:

$$\hat{a}_l = a_l + K \times \frac{m_l}{bw_l}, \quad l \in DIO, \tag{4.8}$$

where K is a positive constant, bw_l is the bandwidth requirement of flow l, and $\hat{a}_l$ is the updated value of a_l.

We use the *absorption* variable to keep track of extra slot usage. If there are any extra slots left by flows in the sets DS and DIU, PFS sorts out flows in the set DIO according to the *absorption* variable. Flows with smaller *absorption* values have a higher priority to use extra slots to transmit more packets. After packet transmissions, a flow's *absorption* variable is increased and the flow has less chance to use any more slots. In the long run, the differences among *absorption* variables are much smaller than variables themselves.

The complete description of the PFS scheduling algorithm is as follows:

Step 1: Pre-calculate the ration for each flow.

Step 2: Initialize grant variable g and absorption variable a to zero for each flow

and add them to the corresponding lists of $\{g_i\}$ and $\{a_i\}$.

Step 3: Start a new frame.

Step 4: Inspect each flow queue, classify flows into three types of DS, DIU, and DIO, update the grant list $\{g_j\}$, and calculate the extra slot es as follows.

4.1: For each DI flow j, if $n_j > r_j, j \in DIO$; Otherwise $j \in DIU, g_j = g_j + (r_j - n_j)$;

4.2: Move g_j toward the beginning of grant list $\{g_j\}$ so that elements in the list $\{g_j\}$ are in descending order;

4.3: $es = \sum_{i \in DS,DI}(N - r_i) + \sum_{j \in DS,DIU}(r_j - n_j) + \sum_{k \in DIO}(r_k - n_k)$.

Step 5: Schedule cells out according to the ration variable r for each flow.

Step 6: Find the first DIO flow identifier k in the grant list $\{g_k\}$, and perform the following operations.

6.1: $m_k = min(es, n_k - r_k), g_k = g_k - m_k, es = es - m_k$;

6.2: Move g_k toward the end of grant list $\{g_k\}$ so that elements in the list $\{g_k\}$ are in descending order;

6.3: If $(es > 0)$, find the next DIO flow identifier until the end of grant list $\{g_k\}$ and repeat the above procedures in the Step 6.

Step 7: Find the first DIO flow identifier k in the absorption list $\{a_k\}$, and perform the following operations.

7.1: $m_k = min(es, u_k), a_k = a_k + K * m_k/bw_k, es = es - m_k$;

7.2: Move a_k toward the end of absorption list $\{a_k\}$ so that elements in the list $\{a_k\}$ are are in ascending order;

7.3 If ($es > 0$), find the next DIO flow identifier until the end of absorption list $\{a_k\}$ and repeat the above procedures in the Step 7.

Step 8: Go back to the Step 3.

In summary, per-flow scheduler (PFS) schedules packet transmission of both delay-sensitive and delay-insensitive flows on a frame basis. For delay-sensitive flows, enough slots are reserved to accommodate peak-rate allocation. Any unused slots will be *immediately* exploited to serve other flows. For delay-insensitive flows, PFS first selects packets under flows' rations. It then allocates any extra slots available to schedule packet transmission for flows that have unused rations. Finally, PFS checks the *absorption* variables to dispatch more packets than flows' rations. PFS operates on the packet level to enforce bandwidth requirement on the flow level. In essence, bandwidth is guaranteed for *all* flows, i.e., both delay-sensitive and delay-insensitive flows. For a delay-insensitive flow, it could achieve a higher throughput than the average bandwidth it requested during the flow's setup time. This feature is well suited to bursty data traffic, since it provides a fast draining mechanism when bursts arrive.

4.4 Performance Analysis

In this section, we conduct performance analysis focusing on the most important QoS parameters: delay, delay jitter, throughput, and buffer requirement. For delay-sensitive traffic, service guarantee is provided for delay, delay jitter, and throughput. For delay-

insensitive traffic, service guarantee is provided only for throughput. Analysis of buffer requirement is used to evaluate the performance of our proposed per-flow queueing and VC-merging switches.

It is assumed that per-flow queueing is implemented in the access switches and VC-merging is implemented in the core switches. A flow has to specify or negotiate a rate during the flow's setup time and call admission control is executed before a flow is admitted. As outlined in [80], delay-sensitive flows use peak-rate in their bandwidth specification, while delay-insensitive flows use average-rate in their bandwidth specification. To ensure stability, it is assumed that the link capacity is greater than the sum of rates of admitted flows, i.e., $C_L > \sum_i bw_i$. Load conditions of system can be stringent as long as the above condition is not violated.

4.4.1 Delay and delay jitter bounds

In a previous section, we proposed a logical frame structure of N packet slots over a link with capacity C_L. The maximum frame duration is denoted as T_{max}. Now we derive the end-to-end delay bound for delay-sensitive flows, assuming the same N-slot frame structure and link capacity C_L within the network. The end-to-end delay is defined as the time elapsed between the arrival of a packet at the first-hop access router (Access Router S in Fig. 4.2) and the receipt of the packet at the last-hop access router (Access Router D in Fig. 4.2).

First we obtain the delay bound introduced by the first core router within the backbone cloud shown in Fig. 4.1. Since we use VC merging in the core routers, multiple IP flows

from access routers are aggregated into a single flow coming out of the first core router. We assume that k delay-sensitive flows are merged in the first core router and the peak-rate summation of these k flows is less than or equal to C_L/N, i.e.,

$$\sum_{i=1}^{k} pr_i \leq \frac{C_L}{N}, \tag{4.9}$$

where pr_i is the peak rate for flow i. Due to VC merging, we arrange packets from these k flows to share the same slot position in consecutive frames leaving out of the first core router.

There are two components for the delay encountered by a packet entering the first core router of a backbone cloud: the waiting time to transmit the packet, and the service time of the packet. We first calculate the waiting time. In the worst case, one packet from each of the k flows arrives at the same time and misses the start of a frame. The packets from k flows have to wait for the start of the next frame. We identify the packet from one flow as "tagged". This "tagged" packet has to wait for the finish of transmission of the packets from other $(k-1)$ flows, and therefore the "tagged" packet experiences the longest waiting time, which is

$$d_w \leq T_{max} + (k-1)T_{max} + T_{max} = (k+1)T_{max}. \tag{4.10}$$

The first term in the above equation represents the upper bound of waiting time for the start of a new frame, the second term is the time elapsed to transmit $(k-1)$ packets in the same slot position, and the last term is the upper bound between the start of frame and the start of transmission of the "tagged" packet.

Now we calculate the second component of the delay which is the packet transmission time. Since we have an N-slot frame and we arrange k flows to share the same slot in consecutive frames, the bandwidth perceived by each of the k flows is $\frac{1}{k} \times \frac{C_L}{N}$. Let l_{max} and l_{min} denote the maximum length and minimum length of packets. The packet transmission time m is

$$m \leq l_{max}/(\frac{1}{k} \times \frac{C_L}{N}) = l_{max} \times \frac{kN}{C_L}. \tag{4.11}$$

Combining the above two terms, we can express the delay d_{1c} at the first core router as

$$d_{1c} = d_w + m \leq (k+1)T_{max} + l_{max} \times \frac{kN}{C_L}. \tag{4.12}$$

Since the above k flows are merged into a single flow leaving out of the first core router and going into other core routers, reusing Eqn. (4.10), we have delays introduced at the other core routers with a maximum of $2T_{max}$ each. If there are h hops between the first core router and the last core router, the maximum delay introduced by the switching routers other than the first one will be $h \times 2T_{max}$. The delay introduced by all core routers is given by

$$d_c = d_{1c} + h \times 2T_{max} \leq (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + h \times 2T_{max}. \tag{4.13}$$

In a similar fashion, we can obtain the delay introduced by the first-hop and the last-hop access routers:

$$d_a \leq 2T_{max} + 2T_{max} = 4T_{max}. \tag{4.14}$$

Adding the propagation delay *prop*, we will attain the end-to-end delay d,

$$d = prop + d_a + d_c \leq prop + (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max}. \quad (4.15)$$

Based on the above analysis and derivation, we can easily obtain the upper bound and lower bound of the end-to-end delay d:

$$d_{max} \leq prop + (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max} \quad (4.16)$$

and

$$d_{min} \geq prop + l_{min} \times \frac{kN}{C_L}. \quad (4.17)$$

The delay jitter dj is defined as the difference between d_{max} and d_{min}. The upper bound for delay jitter can be derived from Eqn. (4.16) and Eqn. (4.17) as follows:

$$\begin{aligned} dj &= d_{max} - d_{min} \\ &\leq [prop + (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max}] - [prop + l_{min} \times \frac{kN}{C_L}] \\ &\leq (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max}. \end{aligned} \quad (4.18)$$

It is straightforward to show that the propagation delay across a wide area (e.g. 50 ms across continental USA) is the dominant factor in the end-to-end delay over a Gb/s or Tb/s network. A significant advantage of introducing an N-slot frame structure is to ensure a low maximum delay in comparison with the end-to-end propagation delay for high-speed

transport.

If the combined rate of aggregated flows is so large that at least two packets have to be forwarded in a frame, we can decompose the flow into sub-flows, such that we can use the above derivation to obtain similar results.

4.4.2 Throughput

Per-flow scheduler (PFS) has attractive properties and they can be expressed by the following lemmas and theorem.

Lemma 1: PFS provides bandwidth guarantee for every flow.

Proof: For an outgoing link with capacity C_L and N (logical) slots in a frame, we allocate slot ration r_i for flow i according to Eqn. (4.1):

$$r_i = \frac{bw_i}{C_L} \times N, \quad i \in \{DS, DI\}. \tag{4.19}$$

The bandwidth allocated by PFS to flow i is

$$\begin{aligned} bw &= \frac{r_i}{N} \times C_L \\ &= \frac{bw_i}{C_L} \times N \times \frac{1}{N} \times C_L \\ &= bw_i \qquad \textbf{Q.E.D.} \end{aligned}$$

Lemma 2: PFS is fair in the sense that it allocates any extra bandwidth in proportion to requested average rates of delay-insensitive flows.

Proof: Let I and J represent extra slots consumed by delay-insensitive flows i and j during the interval $(0, T_l)$. Ignoring the small difference between *absorption* variables a_i and a_j, we have

$$\begin{aligned} a_i &= a_j, \\ K \times \frac{I}{bw_i} &= K \times \frac{J}{bw_j}, \\ \frac{I/T_l}{J/T_l} &= \frac{bw_i}{bw_j}. \end{aligned}$$

Q.E.D.

Theorem: For delay-insensitive flow i, the extra bandwidth received beyond the requested average bandwidth bw_i is

$$\frac{bw_i}{\sum_i bw_i} \times (C_L - \sum_i bw_i - \sum_j bw_j), \quad i \in DI, \ \ j \in DS.$$

Proof: From **Lemma 2**, any bandwidth capacity left, which is $C_L - \sum_i bw_i - \sum_j bw_j$, $i \in DI$, $j \in DS$, will be distributed in proportion to delay-insensitive flows' required average rates. For delay-insensitive flow i which has requested an average bandwidth bw_i, its portion in total required average bandwidth is $bw_i / \sum_i bw_i$, therefore flow i can benefit from the following extra bandwidth beyond bw_i:

$$extra\ \ bandwidth = \frac{bw_i}{\sum_i bw_i} \times (C_L - \sum_i bw_i - \sum_j bw_j), \quad i \in DI, \ \ j \in DS. \qquad \textbf{Q.E.D.}$$

4.4.3 Buffer Occupancy

In this section, we conduct performance analysis of per-flow queueing switches and VC-merging switches. We focus our analysis on the distribution of buffer occupancy. Results indicate that buffer requirement can be excessive for VC merging when compared to non-VC merging. This is in contrast to the main result in [65].

To analyze the buffer requirement for VC merging, we need to characterize the packet arrival process from all input ports. We assume that an ON period corresponds to a single packet, and therefore all the cells within an ON period come from this single packet. As reasoned before, packet transmission to an output port can start only after the last cell of the packet is received because of the non-interleaving requirement of VC merging. In other words, a batch of cells received during the ON period have to be stored in the buffer before being sent out to an output port. It is easy to see that buffer occupancy is affected by these batches of cells. Here comes our key observation: the ON-OFF bursty arrival process can be treated as a bulk arrival process with the bulk size being equal to the corresponding ON period and the bulk interarrival time being equal to the sum of the OFF and ON periods. Embedded Markov chain analysis can be applied to the above bulk arrival process. However, an exact construction of the Markov chain requires an infinite number of states. To simplify the analysis, we approximate the bulk arrival process by a bulk Bernoulli arrival process, i.e., in an any given time slot, the probability that a bulk arrives on a particular input port is p. This probability p can be estimated by the

occurrences of a new start of ON period during ON-OFF cycles, which can be obtained as:

$$p = \frac{1}{\frac{1}{r} + \frac{1}{s}} = \frac{rs}{r+s}. \tag{4.20}$$

The bulk size g is equal to the ON period and therefore it follows a geometric distribution with the following PGF:

$$G(z) = \frac{rz}{1-(1-r)z}. \tag{4.21}$$

To have a fair comparison with per-flow queueing (non-VC merging), we use the same assumption that the buffer size is infinite and each incoming bulk has equal probability $1/N$ of being addressed to any given output port.

Now we use a similar analysis approach as before to derive the queue length distribution for each output port. Fixing our attention on a particular output queue (the "tagged" queue), we define the random variable v as the number of bulk arrivals at the tagged queue during a given time slot. It follows that v has a binomial distribution with probability generating function (PGF)

$$V(z) = \sum_{i=0}^{N} z^i Pr[v=i] = (1 - p/N + zp/N)^N. \tag{4.22}$$

Letting $A(z)$ be the PGF of the number of cell arrivals during a given time slot, then from the above $V(z)$, we have

$$A(z) = V(G(z)) = (1 - p/N + G(z)p/N)^N, \tag{4.23}$$

where $G(z)$ is the PGF of bulk size distribution which is expressed in Eqn. (4.21).

Letting q_m be the number of cells in an output queue at the end of mth time slot, and a_m be the the number of cell arrivals during the mth time slot, then we have

$$q_m = \max(0, \quad q_{m-1} + a_m - 1). \tag{4.24}$$

Using the same technique as before, we obtain the following PGF for the steady-state queue size

$$Q(z) = \frac{(1-\rho)(1-z)}{(1-p/N+G(z)p/N)^N - z}. \tag{4.25}$$

Differentiating Eqn. (4.25) with respect to z and taking the limit as $z \to 1$, we can obtain the mean $\overline{q}$ and variance σ_q^2 of steady-state queue size.

As derived before, the minimum buffer size B_{min} to achieve a desired cell loss probability 10^{-k} is governed by

$$B_{min} = u_k\sqrt{N}\sigma_q + N\overline{q}, \tag{4.26}$$

where the values of $u_k, k = 1, 2, \ldots$, are tabulated in [33].

Figure 4.6 compares the performance of cell loss probability between per-flow queueing (non-VC merging) and VC merging for switch sizes $N = 8$ and 16 under bursty traffic. We select parameters s and r of the ON/OFF model such that they represent both bursty traffic source and heavily offered load. The state transition probability r from ON to OFF state is chosen to be 0.1, and the state transition probability s from OFF to ON state is chosen to be 0.5. The average burst length $1/r$ is 10 cells, and the average idle period

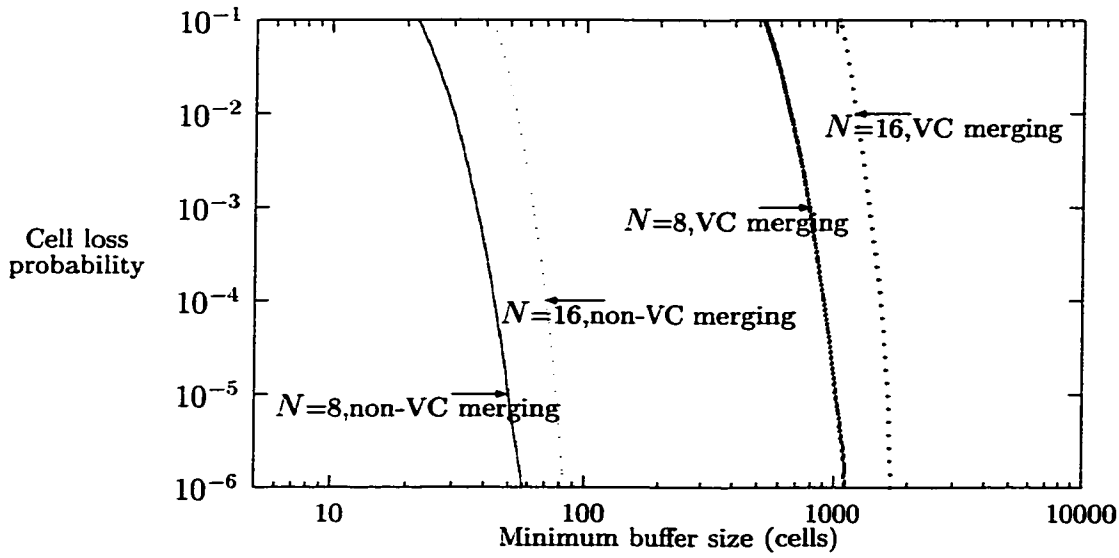

Figure 4.6: Comparison of cell loss probabilities between VC merging and non-VC merging.

$1/s$ is 2 cells. The offered load $\rho = (1/r)/(1/r + 1/s)$ is 0.83, which is high. We can see that the above traffic is quite bursty and heavily loaded. Figure 4.6 clearly indicates that VC merging requires a significant increase in buffer size in order to achieve the same cell loss probability as non-VC merging. The main reason of this excessiveness is due to the fact that it is not work-conserving to do VC merging in a core switch. As reasoned before, packet transmission can not start until the last cell of the packet is received. If the incoming traffic is heavy and bursty, which is exactly the case in Figure 4.6, there are periods in which no packet is transmitted while cells keep coming into the buffer. It will eventually lead to a huge backlog of partial packets, causing temporary congestion inside the buffer. This is the root cause of excessive buffer requirement for VC merging when compared to non-VC merging. The findings presented here are in contrast to a previous

investigation [65].

4.4.4 Discussion

In summary, the switching and scheduling mechanisms in the QoS-capable routers are based on their counterparts in the ATM switch design. Per-VC queueing provides a solid foundation for integrated cell and frame switching and VC merging, while the philosophy in the design of PVQS is extended to IP router design and PFS is developed. From the operation point of view, if we imagine that every packet slot in the logical N-slot frame is fixed to accommodate the maximum-length packet, we immediately establish the one-to-one mapping between the cell slot in the previous chapter and the packet slot in this chapter. There is no change in the simulation programs although the time scales are different because of different context. Fixed-length slot for each packet is the worst scenario, since not all packets reach the maximum length and therefore savings can be easily obtained from switching and scheduling these non-maximum-length packets. For the reason of brevity, simulation results are not shown here since they are very similar to the ones presented in the last chapter.

Each slot in an N-slot frame is designed to accommodate the largest packet. After a packet is sent out, there is no need to wait for the start of next slot in the practical implementation. PFS scheduler can immediately begin sending out the next packet. If there is spare capacity left in a frame, PFS scheduler may choose to sent out more BE packets. The above slot management technique is illustrated in the Figure 4.7.

Figure 4.7 (a) shows that a frame is designed to send out four packets (one DS packet,

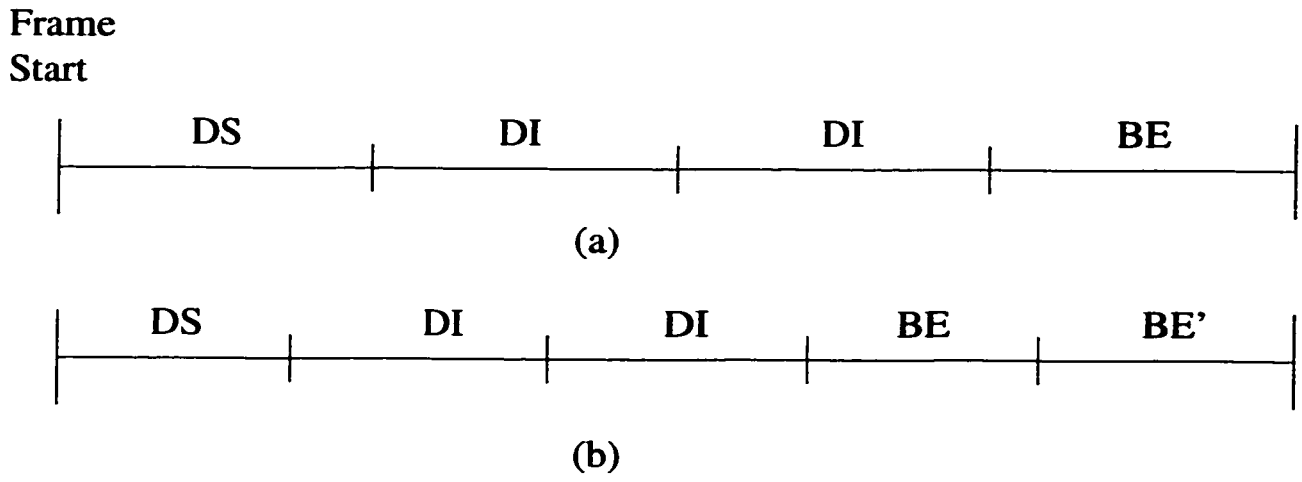

Figure 4.7: Slot management to fully utilize bandwidth capacity.

two DI packets, and one BE packets) at their maximum lengths. After sending out the DS packet which has a higher priority than other packets, PFS scheduler immediately send out the next DI packet. So there is no transmission gap between the end of DS packet and the beginning of DI packet. After all four packets have been sent out, it is quite possible that there is still spare capacity in the frame since not all packets reach their maximum lengths. To take advantage of this available bandwidth resource, PFS scheduler may choose to send out another BE packet (denoted as BE' in the Figure 4.7 (b)) in order to fully utilize bandwidth capacity.

4.5 Reducing Buffer Requirement for VC Merging

Studies in the previous section indicate that the buffer requirement can be excessive for VC merging when compared to non-VC merging. As analyzed before, buffer overloading in

the VC-merging case is a direct result of temporary congestion inside the buffer. Once we obtain an insight to the root cause of a problem, various solutions can be easily found. In principle, any congestion control technique can be applied to avoid or to alleviate congestion if it has occurred. In this section, we choose two effective congestion control methods to significantly reduce the buffer requirement for VC merging. A third alternative of using faster outgoing links was presented in [81].

4.5.1 Flow control

An effective way to avoid congestion is to adopt flow control. A switch needs to collect information about congestion and to inform the traffic sources. This congestion indication is usually based on the amount of buffer space available (or in use) in the switch. Traffic sources react to the congestion feedback information by limiting the amount of traffic they send. This feedback control loop has a delay of at least twice the propagation delay between the switch and the control point. The control loop delay needs to be minimized, since the switch has to buffer all the transit cells that arrive after the switch signals its congestion status but before the end of control loop delay. Based on the above analysis, we adopt a hop-by-hop credit-based flow control scheme [43, 78], rather than an end-to-end rated-based method [21]. Hop-by-hop control loops are much faster than end-to-end ones because of much shorter round-trip time.

4.5.2 Random early detection (RED)

The widespread use of World Wide Web fuels the growth of the Internet traffic. Clicking on a web page can trigger several TCP sessions simultaneously. It has been shown that TCP traffic constitutes about 90 percent of the traffic on the Internet. Since TCP traffic is so prevalent, we assume in this subsection that all incoming traffic is TCP.

TCP aggressively increases its throughput rate until it experiences a drop, and then it tries to maintain a rate as high as possible with a requirement to maintain a low drop rate. In some senses, TCP *provokes* congestion in its search for maximum throughput rate. Therefore, some portion of TCP is guaranteed to be dropped. A new approach to TCP drops, called Random Early Detection (RED) [20], is starting to be deployed on the Internet. RED detects the onset of congestion by monitoring the length of multiple queues and randomly drops only one packet with uniform distribution from each of the high-rate TCP sessions to prevent session stalling.

4.5.3 Simulation Results

In this section, we use simulations to verify the effectiveness of the flow control and RED techniques to reduce buffer requirement of VC merging. To make our simulations more realistic, we only keep the ON-OFF modeling for traffic source and relax all the other assumptions used in the previous analysis. The simulation platform and convergence criteria are the same as those simulations in Chapter 3.

We use simulations in Figure 4.8 to compare the buffer requirement using credit-based

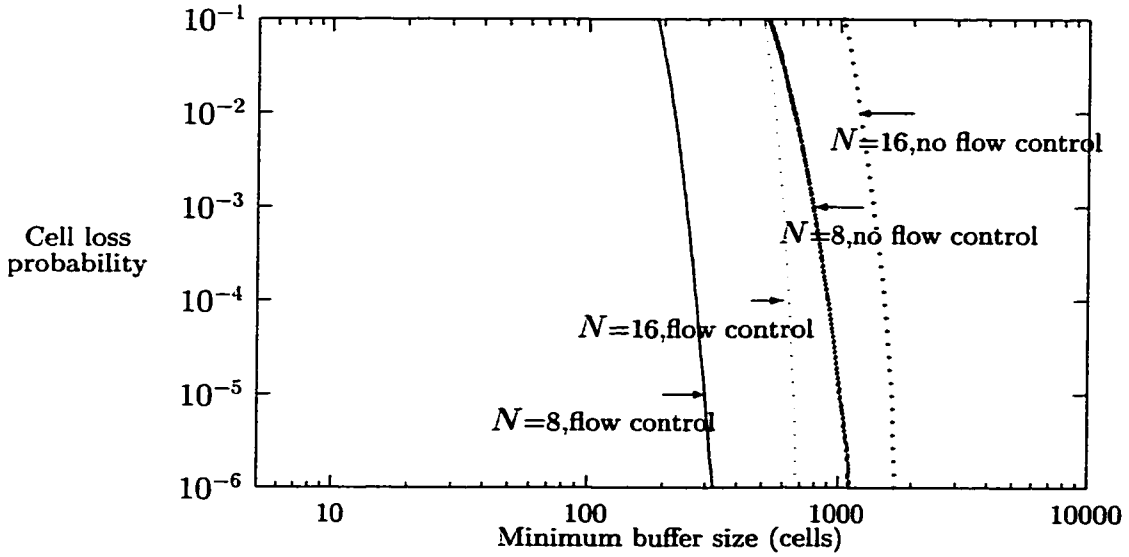

Figure 4.8: Reduction of buffer requirement by using credit-based flow control.

flow control in VC merging against conventional VC merging without any flow control. The switch sizes are chosen to be 8 and 16, and we use the same bursty traffic sources as before. The queue length at each output port is used as a measurement of congestion. The thresholds are chosen to be 32 and 64 for switch sizes $N = 8$ and 16 respectively. The comparison plot is shown in Figure 4.8. In Figure 4.8, the two curves marked with "no flow control" correspond to the conventional VC merging case, and they are the same as the ones which are marked with "VC merging" in the Figure 4.6. From the plot, it is clear that credit-based flow control can significantly reduce the buffer requirement for VC-merge switches. In credit-controlled networks, congestion produces back-pressure, which results in a reduction of credits passed to the upstream switches. The amount of traffic from upstream switches is based on the available credits and it is reduced accordingly. This relieves the congestion experienced by the downstream switches and reduces the buffer

requirement in the downstream switches as witnessed in Figure 4.8.

To verify the effectiveness of the RED scheme, we employ the RED technique in another simulation to compare the buffer requirement using RED in VC merging against conventional VC merging without packet discard. The plot of comparison is shown in Figure 4.9. As usual, we use the queue length of each output port as an indication of congestion. The thresholds are chosen to be 32 and 64 for switch sizes $N = 8$ and 16 respectively. Once the onset of congestion is detected, only one packet from one VC is dropped as in RED. In Figure 4.9, the two curves marked with "no RED" correspond to the conventional VC merging case, and they are the same as the ones which are marked with "VC merging" in the Figure 4.6. From Figure 4.9, we can clearly notice that the RED technique leads to a significant decrease in terms of buffer occupancy for both switch sizes $N = 8$ and 16.

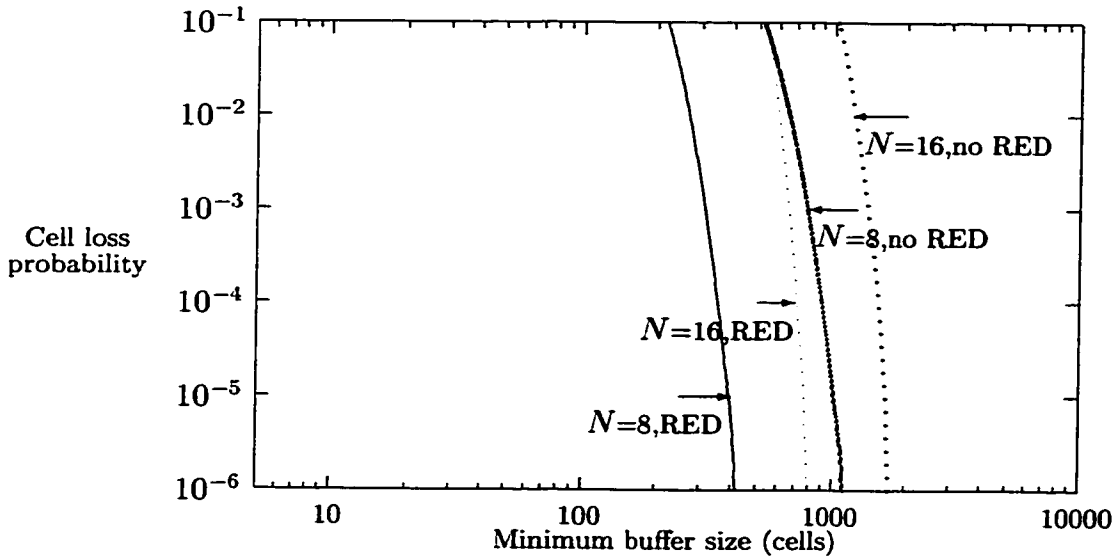

Figure 4.9: Reduction of buffer requirement by using Random Early Detection (RED).

4.6 Comparison with other IP QoS Effort

The original TCP/IP protocol suite was designed to support only service class: best effort. Therefore TCP/IP networks would make their best effort to deliver packets to their destinations but with no guarantees and no special resources allocated for any of the packets. As the demands for real-time applications grow, it becomes obvious that best-effort service is inadequate to support them. In response to the need for IP QoS, IETF started to define an Integrated Service Architecture (IntServ) [7] that would extend existing IP architecture model to support both real-time and best-effort traffic. The signaling protocol used by the IntServ was the Resource Reservation Protocol, otherwise known as RSVP. To support IP QoS, IntServ/RSVP requires installation and maintenance of per-flow state in routers. This severely limited IntServ/RSVP's applicability and scalability over large networks [48]. Simply put, it is impossible to maintain and refresh state information for each of a huge number of traffic flows in the core of large networks.

Recognizing the shortcomings of the fine-grained, per-flow approach of IntServ/RSVP, IETF proposed a Differentiated Services (DiffServ) [4] as a means of providing a scalable and coarse level of service suitable for large networks. There are two key components in the DiffServ architecture: DS-field and per-hop behavior (PHB). The DS-field is a bit pattern in the IP header of each packet that denotes the service (termed as per-hop behavior or PHB) the packet should receive at each hop in an end-to-end path. The per-hop behavior (PHB) defines the service th packet receives at each hop as it is forwarded through the network. As the name of "Differentiated Services" implies, DiffServ provides a framework

to treat one flow better than some other flows, so it is a soft guarantee. The current view is that DiffServ along with more bandwidth will enable the Internet to offer QoS [49].

In this thesis, we follow basic principles of DiffServ of marking traffic flows at the access routers and aggregate a number of traffic flows into a single flow in the core routers. DiffServ recommends *what* should be done. This thesis presents a new way on *how* it can be implemented. In particular, our studies show that per-VC queueing architecture is instrumental to the design of per-flow queueing in access routers and VC-merging in core routers. In conjunction with the scheduling algorithm (PFS), hard QoS guarantee, in terms of delay, delay jitter, and throughput, can be provided for IP flows. However, we use an overlay model to map IP traffic flows over ATM VCs. This limits the applicability of our scheme to only those networks with underlying ATM support.

4.7 Concluding Remarks

In this chapter, we presented our insight to the issue of providing QoS guarantee in IP over ATM networks. Switching and scheduling were still the key ingredients in IP QoS schemes. Based on ATM's per-VC queueing architecture, we introduced integrated frame and cell switching. On the scheduling front, we extended Per-VC Queueing Scheduler (PVQS) and presented per-flow scheduling (PFS).

Chapter 5

Design Guideline

The purpose of this chapter is to recommend design procedures when implementing switching architectures and scheduling algorithms in ATM and IP networks.

Figure 5.1 reflects the flow chart for designing per-VC queueing ATM switches. There are two independent parameter, the switch size N and the number of VC connections M, to be chosen at the beginning. Therefore two branches are forked at the start and they are merged before proceeding to pick a scheduling algorithm. Any scheduling mechanism can be chosen, but the PVQS is a preferred choice since it takes advantage of per-VC queueing.

The design of IP switching routers are based upon the architecture of per-VC queueing. The design procedure is very similar except that Figure 4.6 should be consulted before deciding the minimum buffer size.

Figure 5.2 reflects the flow chart for designing per-VC queueing scheduler (PVQS). At the start, the frame size N needs to be decided. Using Eqn. (3.12) in Chapter 3, one should check if the maximum queueing at the scheduler meets the expectation. If so, it

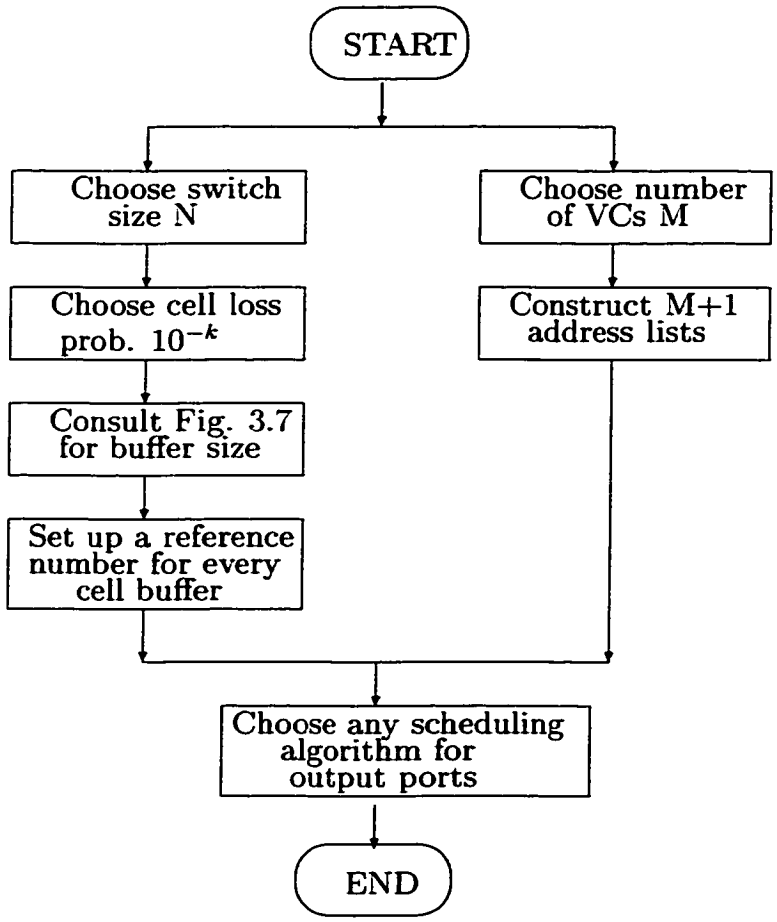

Figure 5.1: Design procedure for per-VC queueing ATM switch.

proceeds to the next step of choosing slot ration for each connection. If not, one needs to reduce the frame size (say by 10%) to reduce the queueing delay. In the high-speed network, queueing delay at the node is minimal when compared with propagation delay. Even a large N usually will not cause much queueing delay at the scheduler.

The design of per-flow scheduling (PFS) is based upon the per-VC queueing scheduler (PVQS). The design procedure is very similar except that the slot is used to accommodate an IP packet instead of an ATM cell.

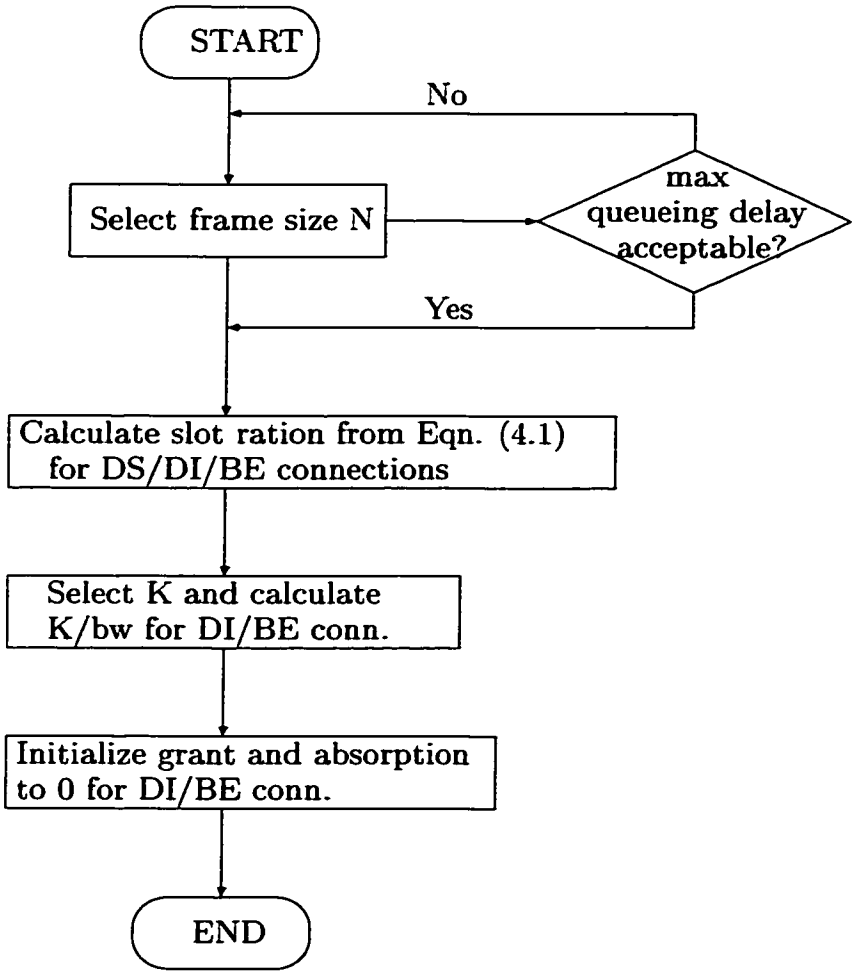

Figure 5.2: Design procedure for per-VC queueing scheduler.

Chapter 6

Conclusions

In this thesis, we focused on the two key elements in any traffic scheme for ATM/IP networks: switching and scheduling. We undertook extensive literature review, recognized various issues from our studies, and proposed remedies for existing work. We first developed switching architecture and scheduling schemes theoretically, then developed formulas to evaluate the performance of proposed schemes, and finally performed simulations to validate the analytical results.

In this thesis, we simplified traffic characterization and introduced a classification method based on the delay attribute. We introduced innovative switching architecture and scheduling algorithm designed to keep them in line with network operations. In ATM networks, we studied per-VC queueing architecture and per-VC scheduling to take full advantage of per-VC queueing. In IP over ATM networks, we studied per-flow queueing in access routers and VC-merge in core routers to address the scalability issue. We also introduced per-flow scheduling in tandem to our switching architecture. We conducted

performance analysis to evaluate the performance of our proposed switching and scheduling schemes. Simulations were used to validate our analytical results. Therefore we have successfully fulfilled our original objectives, which were to seek alternatives in traffic classification, switching, and scheduling in order to address the shortcomings in the above research areas.

As an evolution of our research in the key area of switching and scheduling, we shall take a system-wide approach and build a comprehensive framework on traffic control, aiming to examine various components of traffic control and explore the interactions among them.

Bibliography

[1] H. Ahmadi and W. Denzel, "A survey of modern high-performance switching techniques," *IEEE J. Select. Areas Commun.*, vol. 7, no. 7, pp. 1091-1102, Sept. 1989.

[2] R. Awdeh and H. Mouftah, "Survey of ATM switch architectures," *Computer Networks and ISDN Systems,* vol. 27, pp. 1567-1613, Nov. 1995.

[3] J. Bennett and H. Zhang, "WF2Q: worst-case fair weight fair queueing," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996, pp. 120-128.

[4] S. Blake, et al, "A Framework for differentiated services," Internet draft, *draft-ietf-diffserv-framework-01.txt*, Oct. 1998.

[5] F. Bonomi and K. Fendick, "The rate-based flow control framework for the available bit rate ATM service," *IEEE Network Magazine*, vol. 9, no. 2, pp. 25-39, Mar./Apr. 1995.

[6] R. Braden, "Requirements for Internet hosts - communication layers," *RFC 1122*, Oct. 1989.

[7] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," *RFC 1633*, Jun. 1994.

[8] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan, "A framework for Multiprotocol Label Switching," Internet draft, *draft-ietf-mpls-framework-00.txt,* May 1997.

[9] F. Chiussi, "Implementing fair queueing in ATM networks - part 2: the logarithmic calendar queue," in *Proc. IEEE GLOBECOM'97*, Phoenix, AZ, Nov. 1997, pp. 519-525.

[10] I. Cidon, I. Gopal, and H. Heleis, "PARIS: an approach to integrated private networks," in *Proc. IEEE ICC'87*, Seattle, WA, Jun. 1987, pp. 764-768.

[11] Cisco, "Architecture for voice, video and integrated data," http://www.cisco.com/warp/public/cc/so/neso/vvda/iptl/avvid_wp.htm.

[12] J. Coudreuse and M. Servel, "Prelude: an asynchronous time-division packet switched network," in *Proc. IEEE ICC'87*, Seattle, WA, Jun. 1987, pp. 769-773.

[13] J. Daigle and J. Langform, "Models for analysis of packet-voice communication systems," *IEEE JSAC*, vol. 4, no. 6, pp. 847-855, 1986.

[14] M. De Prycher, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, 3rd edition. Chichester, England: Ellis Horwood Limited, 1995.

[15] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM'89*, Austin, TX, Sept. 1989, pp. 1-12.

[16] A. Elwalid, D. Mitra, and T. Stern, "Statistical multiplexing of Markov modulated sources: theory and computational algorithms,", in *Proc. ITC-13*, 1989, pp 495-500.

[17] A. Elwalid and D. Mitra, "Effective bandwidth of bursty, variable rate sources for admission control to BISDN," in *Proc. IEEE ICC'93,* Geneva, Switzerland, 1993, pp. 1325-1330.

[18] A. Elwalid and D. Mitra, "Effective bandwidth of general markovian traffic sources and admission control of high speed networks," *IEEE/ACM Trans. on Networking,* vol. 1, pp. 329-343, Jun. 1993.

[19] A. Elwalid, D. Mitra, and R. Wentworth, "A new approach for allocating buffers and bandwidth to heterogeneous regulated traffic in an ATM node," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1115-1127, Aug. 1995.

[20] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking,* vol. 1, pp. 397-413, Aug. 1993.

[21] ATM Forum, "ATM Traffic Management Specification," Version 4.0, *af-tm-0056.000*, Apr. 1996.

[22] R. Gibbens and P. Hunt, "Effective bandwidths for the multi-type UAS channel," *Queuing Systems*, vol. 9, pp. 17-27, Oct. 1991.

[23] N. Giroux and S. Ganti, *Quality of service in ATM networks: state-of-the-art traffic management.* Upper Saddle River, NJ: Prentice Hall, 1999.

[24] S. Golestani, A self-clocked fair queueing scheme for broadband applications," in *Proc. IEEE INFOCOM'94*, Toronto, Ontario, Canada, Jun. 1994, pp. 636-646.

[25] R. Guerin, H. Ahmadi, and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high speed networks," *IEEE J. Select. Areas Commun.*, vol. 9, no. 3, pp. 968-81, Apr. 1990.

[26] M. R. Hashemi and A. Leon-Garcia, "The single-queue switch: a building block for switches with programmable scheduling," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 785-794, Jun. 1997.

[27] M. Hluchyj and M. Karol, "Queueing in high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1587-1597, Dec. 1988.

[28] J. Hui, "A broadband packet switch for multi-rate services," in *Proc. IEEE ICC'87*, Seattle, WA, Jun. 1987, pp. 782-788.

[29] IDT77V400, http://www.idt.com.

[30] IBM, "Voice-data integration in e-business," http://www.networking.ibm.com/voice/voice-data.html.

[31] D. LeGall, "MPEG: a video compression standard for multimedia applications," *Commun. of the ACM*, vol. 34, no. 4, pp. 305-313, Apr. 1991.

[32] R. Jain, *The art of computer systems performance analysis.* New York, NY: John Wiley & Sons, Inc., 1991.

[33] A. Leon-Garcia, *Probability and random processes for electrical engineering,* 2nd edition. Reading, Massachusetts: Addison-Wesley, 1994.

[34] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, no. 12, pp. 1347-1356, 1987.

[35] T. Kozaki et al., "32 $\times$ 32 shared buffer type ATM switch VLSIs for B-ISDN,", in *Proc. IEEE ICC'91*, Jun. 91, pp. 711-715.

[36] Y. Katsube, K. Nagami, and H. Esaki, "Toshiba's router architecture extensions for ATM: overview," *RFC 2098*, Toshiba R & D Center, Feb. 1997.

[37] F. Kelly, "Effective bandwidths at multi-class queues," *Queuing Systems*, vol. 9, pp. 5-16, Oct. 1991.

[38] G. Kesidis, J. Walrand, and C. Chang, "Effective bandwidths for multiclass Markov fluids and other ATM sources," *IEEE/ACM Trans. on Networking,* vol. 1, pp. 424-428, Aug. 1993.

[39] L. Kleinrock, *Queueing systems, vol. 1: theory.* New York: Wiley, 1975.

[40] L. Kleinrock, *Queueing Systems, vol. 2: computer application.* New York: Wiley, 1976.

[41] L. Kleinrock, "The latency/bandwidth tradeoff in gigabit networks," *IEEE Commun. Magazine*, vol. 30, no. 4, pp. 36-40, Apr. 1992.

[42] V. Kulkarni, L. Gun, and P. Chimento, "Effective bandwidth vectors for multiclass traffic multiplexed in partitioned buffer," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1039-1047, Aug. 1995.

[43] H. Kung and R. Morris, "Credit-based flow control for ATM networks," *IEEE Network Magazine*, vol. 9, no. 2, pp. 40-48, Mar./Apr. 1995.

[44] L. Lambarelli, "ATM service categories: the benefits to the user," http://www.atmforum.com/atmforum/library/service_categories.html.

[45] C. Lea, "The load sharing Banyan network," *IEEE Trans. Commun.*, vol. 35, no. 12, pp. 1025-1034, Dec. 1986.

[46] W. Leland et al., "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, Jan. 1994.

[47] J. Mackie-Mason and H. Varian, "Pricing the Internet," in *Proc. 2nd Int. Conf. Telecommun. Syst. Modelling, Anal.*, Nashiville, TN, Mar. 1994, pp. 378-393.

[48] A. Mankin ed., "Resource ReSerVation Protocol (RSVP) version 1 applicability statement: some guidelines on deployment," *RFC 2208*, Sept. 1997.

[49] C. Metz, "IP QoS: traveling in the first class on the Internet," *IEEE Internet Computing*, vol. 3, no. 2, Mar./Apr. 1999.

[50] P. Newman, T. Lyon, and G. Minshell, "Flow labelled IP: a connectionless approach to ATM," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996, pp. 1251-1260.

[51] R. Onvural, *Asynchronous Transfer Mode Networks: Performance Issues*, 2nd edition. Norwood, MA: Artech House, 1995.

[52] OPNET: http://www.mil3.com.

[53] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks-the single node case," in *Proc. IEEE INFOCOM'92*, Florence, Italy, May 1992, pp. 915-924.

[54] V. Paxson and S. Floyd, "Wide-area traffic: the failure of Poisson modeling," in *Proc. ACM SIGCOMM'94*, London, UK, 1994, pp. 257-268.

[55] P. Pruthi and A. Erramilli, "Heavy-tailed ON/OFF source behavior and self-similar traffic," in *Proc. IEEE ICC'95*, 1995, pp. 445-450.

[56] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow, "Cisco Systems' tag switching architecture overview," *RFC 2105*, Cisco Systems Inc., Feb. 1997.

[57] J. Roberts, "Virtual spacing for flexible traffic control," *Int. J. of Commun. Systems*, vol. 7, pp. 307-318, 1994.

[58] M. Schwartz, *Broadband integrated networks.* Upper Saddle River, NJ: Prentice Hall, 1996.

[59] S. Shenker, "Fundamental design issues for the future Internet," *IEEE J. Select. Areas Commun.*, vol. 13, no. 7, pp. 1176-1188, Sept. 1995.

[60] K. Siu, "A new data service in ATM?" *IEEE Network Magazine*, vol. 11, no. 4, pp. 4-5, Jul./Aug. 1997.

[61] H. Suzuki, H. Nagano, and T. Suzuki, "Output buffer switch architecture for asynchronous transfer mode," in *Proc. IEEE ICC'89*, Boston, MA, Jun. 1989, pp. 99-103.

[62] T. Takeuchi et al., "Synchronous composite packet switching for B-ISDN," *IEEE J. Select. Areas Commun.*, vol. 5, pp. 1365-1376, 1987.

[63] A. Tanenbaum, *Computer Networks*, 2nd edition. Englewood Cliffs, NJ: Prentice Hall, 1988.

[64] A. Viswanathan, N. Feldman, R. Boivie, and R. Woundy, "ARIS: aggregate route-based IP switching," Internet Draft, *draft-viswanathan-aris-overview-00.txt*, IBM, Mar. 1997.

[65] I. Widjaja and A. Elwalid, "Performance issues in VC-merge capable switches for IP over ATM networks," in *Proc. IEEE INFOCOM'98*, San Francisco, CA, Mar. 1998, pp. 372-380.

[66] G. Woodruff and R. Kositpaiboon, "Multimedia traffic management principles for guaranteed ATM network performance," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 437-446, Apr. 1990.

[67] M. Taqqu and J. Levy, "Using renewal processes to generate long grange dependence and high variability," *Dependence in Probability and Statistics, and Progress in Probability*, vol. 3, pp. 73-89, 1986.

[68] F. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," *Proc. IEEE,* vol. 78, pp. 133-167, Jan. 1990.

[69] W. Willinger, "Traffic modeling of high-speed networks: theory and practice," in *Stochastic Networks*, Kelly and Williams eds, Springer-Verlag, 1994.

[70] Y. Yeh, M. Hluchyj, and A. Acampora, "The Knockout switch: a simple architecture for high performance packet switching," in *Proc. ISS'87*, Mar. 1987.

[71] L. Zhang, "Designing a new architecture for packet switching communication networks," *IEEE Commun. Magazine*, vol. 25, no. 9, pp. 5-12, Sept. 1987.

[72] L. Zhang, "VirtualClock: a new traffic control algorithm for packet switching networks," in *Proc. ACM SIGCOMM'90*, Philadelphia, PA, Sept. 1990, pp. 19-29.

[73] Z. Zhang and A. Acampora, "Equivalent bandwidth for heterogeneous sources in ATM networks," in *Proc. IEEE ICC'94*, New Orleans, LA, May 1994, pp. 1025-1031.

[74] P. Zhou and O. Yang, "Traffic control in high-speed ATM networks," *CCNR technical report, Dept. of Electrical and Computer Engineering, U. of Ottawa, Ottawa, Ontario, Canada, Oct. 1996.* An extended version was presented in *Proc. IC3N'98*, Lafayette, LA, Oct. 1998, pp. 183-190.

[75] P. Zhou and O. Yang, "Integrated cell and frame switching in ATM networks," *CCNR technical report, Dept. of Electrical and Computer Engineering, University of Ottawa, Ottawa, Ontario, Canada, Oct. 1997.* A brief version appeared in *IEEE Commun. Lets.*, vol. 3, no. 6, pp. 183-184, 1999.

[76] P. Zhou and O. Yang, "A new design of central-queueing ATM switches," in *Proc. IEEE GLOBECOM'97*, Phoenix, AZ, Nov. 1997, pp. 541-545.

[77] P. Zhou and O. Yang, "Design of per-VC queueing ATM switches," in *Proc. IEEE ICC'98*, Atlanta, GA, Jun. 1998, pp. 304-308.

[78] P. Zhou and O. Yang, "A framework of flow control in high-speed ATM networks," in *Proc. IEEE MILCOM'98*, Bedford, MA, Oct. 1998, pp. 782-786.

[79] P. Zhou and O. Yang, "A feasible scheduling algorithm for per-VC queueing ATM switches," in *Proc. ICATM'99*, Colmar, France, Jun. 1999, pp. 295-304.

[80] P. Zhou and O. Yang, "Scalability and QoS guarantee in IP networks," in *Proc. IEEE IC3N'99*, Boston, MA, Oct. 1999, pp. 427-433.

[81] P. Zhou and O. Yang, "Reducing buffer requirement for VC-merge capable ATM switches," in *Proc. IEEE GLOBECOM'99*, Rio de Janeiro, Brazil, Dec.1999, pp. 44-48.

[82] S. Jha, P. Zhou, and O. Yang, "Performance evaluation of joint per-VC scheduling and credit-based flow control in high-speed ATM networks," in *Proc. IEEE IPCCC'00*, Phoenix, AZ, Feb. 2000, pp. 571-577.

[83] P. Zhou and O. Yang, "Design and analysis of per-flow queueing switches and VC-merge switches based on the per-VC queueing architecture," to appear in *Computer Commun.*, special issue on support of multimedia communications over the Internet.