

# Bandwidth Scheduling and its Application in ATM Networks

by

Anthony Hung

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Electrical Engineering

Waterloo, Ontario, Canada, 1997

©Anthony Hung 1997





National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file* *Votre référence*

*Our file* *Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-22210-1

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.



## Abstract

The design goals for bandwidth schedulers in ATM switches are presented; one crucial goal is the ability to guarantee service bandwidth to individual queues. A performance criterion which can measure how well schedulers can guarantee bandwidth, called the minimum-bandwidth property, is given and it is derived for a number of well-known schedulers. The minimum-bandwidth property is compared with another service measure called the service curve. Of the two measures, it is shown that the minimum-bandwidth property gives a superior method to determine how a scheduler guarantees bandwidth. The minimum-bandwidth property is useful in the context of a virtual channel within an ATM network because it makes possible end-to-end delay bounds and buffer sizing results, provided that the source is  $(\sigma, \rho)$ -constrained. These results are applied in the case of prerecorded video transmission. Scheduler design goals other than service guarantees include the ability to control the idle bandwidth distribution and implementability at high speeds. The use of Hierarchical Round-Robin scheduling is argued on the grounds that it fulfills all the design goals. Finally, a method of providing bandwidth guarantees using an input-buffered switch, as opposed to an output-buffered switch, is discussed.



## Acknowledgements

I would like to express many thanks to my supervisor Prof. George Kesidis, whom I first began working with as an undergrad in the Combined Bachelor-Masters program. He has been an enormous help to me throughout my four years of research. I also thank Prof. Jon W. Mark for his guidance. Many thanks also go to the committee members, especially Prof. Tiko Kameda, for helping to improve this thesis. I appreciate those at Spar Aerospace who assisted me during my stay: Marie-José Montpetit, Martin Côté and Pierre Coutu. I also thank the following people for their useful comments: Bill Bishop, Shakil Siddique, Mala Francis, Michael Cheung, Dalia Fayek, James Qiu and Nasir Ghani. Finally, I would like to thank Dora and my parents for their help and encouragement.

This research was supported by the University of Waterloo E&CE Department and NSERC of Canada.

Dedicated to the memory of my sister Lucia.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	The Expected Contributions of this Thesis . . . . .	5
1.2	The Organization of this Thesis . . . . .	6
<b>2</b>	<b>The Basic Model and Background Material</b>	<b>7</b>
2.1	Discrete-Time Modeling . . . . .	7
2.2	Fluid Modeling . . . . .	9
2.3	The Lindley Process . . . . .	10
2.4	ATM Service Classes . . . . .	11
2.5	Usage Parameter Control and the $(\sigma, \rho)$ Constraint . . . . .	13
<b>3</b>	<b>The Minimum-Bandwidth Property of Schedulers</b>	<b>17</b>
3.1	PSN Design Goals . . . . .	17
3.2	The Minimum-Bandwidth Property . . . . .	20
3.3	Schedulers with Minimum-Bandwidth Properties . . . . .	23
3.3.1	Work-Conserving VFT Schedulers . . . . .	24
3.3.2	Other Timestamping Schedulers . . . . .	29
3.3.3	Round-Robin Schedulers . . . . .	31
3.3.4	The Leaky Bucket . . . . .	42

3.3.5	Summary . . . . .	42
3.4	The Tightness of the Minimum-Bandwidth Property . . . . .	43
3.5	Schedulers without Minimum-Bandwidth Properties . . . . .	43
3.5.1	The HOL Scheduler . . . . .	44
3.6	Service Curve . . . . .	47
3.7	Idle Bandwidth Distribution . . . . .	51
<b>4</b>	<b>Extending the Minimum-Bandwidth Property</b>	<b>57</b>
4.1	End-to-End Delay Bound . . . . .	57
4.2	Delay Bound and Buffer Sizing for $(\sigma, \rho)$ -Constrained Sources . . . . .	63
4.3	Delay Jitter and $(\sigma, \rho)$ Shaping of Departure Processes . . . . .	68
4.4	Connections within Virtual Paths . . . . .	72
4.4.1	The Single-Node Case . . . . .	73
4.4.2	The Multiple-Node Case . . . . .	77
4.4.3	$(\sigma, \rho)$ Shaping of Departure Processes . . . . .	77
4.5	A Virtual Channel Example . . . . .	79
<b>5</b>	<b>Application: Transmission of Prerecorded Video</b>	<b>83</b>
5.1	Properties of the Prerecorded Video Service . . . . .	84
5.2	Piecewise-CBR . . . . .	86
5.3	Resource Provisioning for the Reference Network . . . . .	87
5.3.1	Approaches to Finding $\bar{\phi}$ . . . . .	92
5.3.2	The Case for Piecewise-CBR . . . . .	94
5.4	Resource Allocation Over a Virtual Channel . . . . .	95
<b>6</b>	<b>Implementation and Complexity Issues</b>	<b>103</b>
6.1	Implementable Scheduling . . . . .	103

6.2	Input-buffering as an Alternative to Output-Buffering . . . . .	113
7	Summary and Conclusions	123
7.1	Summary . . . . .	123
7.2	Conclusions and Future Directions . . . . .	125
7.3	Related Work: ATM via Satellite . . . . .	126
A	Glossary	129
	Bibliography	139



# List of Tables

3.1	A worst-case scenario for SCFQ . . . . .	30
3.2	Comparison of the minimum-bandwidth properties of round-robin sched- ulers . . . . .	41
3.3	Timestamps and service for the HOL scheduler. . . . .	47
6.1	Arbitration schedules . . . . .	121



# List of Figures

1.1	B-ISDN protocol stack . . . . .	2
1.2	Processor sharing node . . . . .	4
2.1	Shortest possible queueing delay . . . . .	8
2.2	Work done in the fluid and discrete-time models . . . . .	9
2.3	A leaky bucket . . . . .	14
2.4	Lossless transmission for a $(\sigma, \rho)$ -constrained arrival process . . . . .	15
3.1	Coupling PSN queue to the reference queue . . . . .	23
3.2	An HRR frame structure example . . . . .	33
3.3	Graphical explanation of Theorem 3.3 . . . . .	40
3.4	The $(\sigma^{svc}, \rho)$ or linear service curve . . . . .	48
3.5	The linear service curve of a minimum-bandwidth scheduler . . . . .	51
3.6	Hierarchical scheduling . . . . .	54
4.1	A virtual channel of tandem queues . . . . .	58
4.2	Coupling tandem PSN queues to a reference queue . . . . .	59
4.3	Coupling a queueing system to a reference queue . . . . .	62
4.4	Buffer size requirement and burstiness of the departure process of a PSN queue . . . . .	66

4.5	Model for the playback buffer . . . . .	70
4.6	Rate-controlled service discipline . . . . .	73
4.7	Connections sharing a VPC queue . . . . .	74
4.8	A VPC of tandem queues . . . . .	78
4.9	A example of feedback when using VPCs . . . . .	79
4.10	An example of a virtual channel . . . . .	81
5.1	Prerecorded video transmission . . . . .	84
5.2	The video virtual channel with a reference network . . . . .	88
5.3	Transmission which avoids buffer starvation and overflow. . . . .	90
5.4	Effect of $\mathcal{D}_{\mathcal{P}}$ on $\phi_1$ . . . . .	93
5.5	MPEG-1 trace of “Star Wars” . . . . .	96
5.6	The virtual channel for video transmission . . . . .	97
6.1	An architecture of a PSN . . . . .	104
6.2	A sequencer . . . . .	106
6.3	HRR frame structure after three connections are established . . . . .	110
6.4	HRR frame structure after the second connection is terminated . . . . .	111
6.5	The three different types of round-robin scheduling . . . . .	112
6.6	HRR frame structure using semi-idling after second connection is terminated	112
6.7	The clumping effect of idling (a) and semi-idling (b) HRR . . . . .	114
6.8	The worst (a) and best (b) case for routing . . . . .	115
6.9	Comparison of input and output-buffering . . . . .	117
6.10	An input-buffered switch . . . . .	119

## Abbreviations

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ABT	ATM Block Transfer
ACR	Allowed Cell Rate
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CBR	Constant Bit Rate
FIFO	First-in-first-out
Gbps	Gigabits per second
GCRA	Generic Cell Rate Algorithm
GPS	Generalized Processor Sharing
HOL	Head-of-line
HRR	Hierarchal Round-Robin (scheduler)
ITU	International Telecommunications Union
kbps	Kilobits per second
Mbps	Megabits per second
MBS	Maximum Burst Size
MCR	Minimum Cell Rate

MPEG	Moving Pictures Experts Group
PCR	Peak Cell Rate
PGPS	Packetized Generalized Processor Sharing
PSN	Processor sharing node
QoS	Quality of Service
RM	Resource Management
SAP	Service Access Point
SCFQ	Self-Clocked Fair Queueing (scheduler)
SCR	Sustainable Cell Rate
TDM	Time division multiplexing
TST	Time-space-time division multiplexing
UBR	Unspecified Bit Rate
UNI	User network interface
UPC	Usage parameter control
VBR	Variable Bit Rate
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VC-VFT	VirtualClock-Virtual Finishing Time
VFT	Virtual Finishing Time

VPC    Virtual Path Connection  
VPI    Virtual Path Identifier  
WRR    Weighted Round-Robin (scheduler)



# Chapter 1

## Introduction

The world of digital communications has been moving forward at a fast pace in the last two decades. Initially, the links over which data was transferred were slow, of low quality and costly; therefore, the variety of network applications were limited. Over time, however, communications technology improved. It is now at the point where links can offer extremely high rates reliably, and more and more people are able to obtain access to such technology. These factors make it possible now for the communications providers to offer services that were not feasible before, as well as making existing services cheaper. As a result, the communications community has reached a general consensus that now is the time to implement a standard network, called the Broadband Integrated Services Digital Network (B-ISDN), which can service all of the different communications needs of its users. This network can support traditional applications such as voice and data, as well as a wealth of newer ones such as multimedia communications, video-on-demand, etc.

The widely accepted method of data transfer for B-ISDN is the Asynchronous Transfer Mode (ATM) [13] [29]. ATM was first recommended as a standard in 1988, and since then an organization (ATM Forum) has been formed to recommend ATM standards

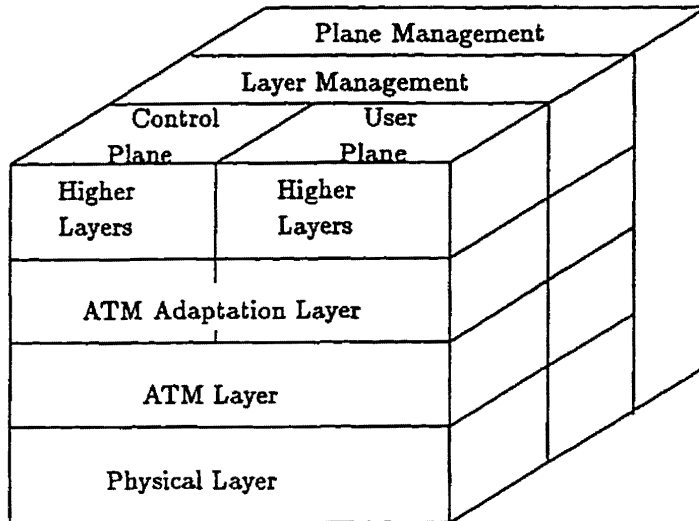


Figure 1.1: B-ISDN protocol stack

to the International Telecommunications Union (ITU). The transfer mode of a network covers aspects related to transmission, multiplexing and switching. An ATM network is a connection oriented, fast packet switching network. A connection, called a Virtual Channel Connection (VCC), is given a fixed path called a *virtual channel* (or commonly referred to as a virtual circuit), and its packets are called cells. Cells are of a fixed length of 53 bytes with 48 bytes of payload and 5 bytes of header. ATM is, for the most part, intended to be a point-to-point switching network employing fiber optics technology.

The three-dimensional B-ISDN protocol stack is shown in Figure 1.1 [18]. User data in the B-ISDN network travels through Service Access Points (SAPs) in the User Plane. Signaling messages pass through SAPs in the Control Plane. Signaling messages are for matters such as the establishment and maintenance of connections.

The main focus of this thesis is the design of the *bandwidth scheduling* algorithms within ATM switches. Simply stated, a bandwidth scheduler creates an ordering of cells

for transmission. It is critical that the schedulers be designed in such a way that the user's required quality of service (QoS) can be achieved. Obviously, matters relating to QoS refer to the ability of the network to satisfy the transmission requirements of the user's data cells. Therefore, this thesis is concerned with the transmission of user data flowing through the User Plane. This thesis is not concerned with the protocol issues of the higher layers and the ATM Adaptation Layers (AALs). When considering the effect of protocol overhead on performance, it is (reasonably) assumed that the higher layers and the AAL incur a constant and relatively small delay at the edges of the ATM network. The part of the protocol stack that this thesis deals with the most is the ATM layer User Plane. This portion of the protocol stack is responsible for translation of the Virtual Channel Identifier (VCI) and Virtual Path Identifier (VPI), multiplexing and demultiplexing function at the switches, flow control for Available Bit Rate (ABR) service, etc.

The fundamental building block used throughout the thesis is the processor sharing node (PSN). The PSN model can be naturally associated with the output port of an *output-buffered* switch, which is shown in Figure 1.2. The figure shows that the PSN takes, at its input, a stream of cells from the switch fabric. In an output-buffered switch, this cell stream is composed of cells from all the input ports which are destined to the particular output port. Upon entry to the PSN, a cell is put into its first-in-first-out (FIFO) queue. While awaiting service, the cell contends for service (transmission) with cells from other queues; a scheduler in the PSN resolves this contention by creating an order of service.

In an output-buffered switch, a cell arriving at an input port is immediately routed to an output port before another cell arrives at that input port; therefore, input-buffering is not necessary. Of course, cells must be stored in output-buffered queues due to possible contention for the output link with other cells. Output-buffering is the optimal queueing

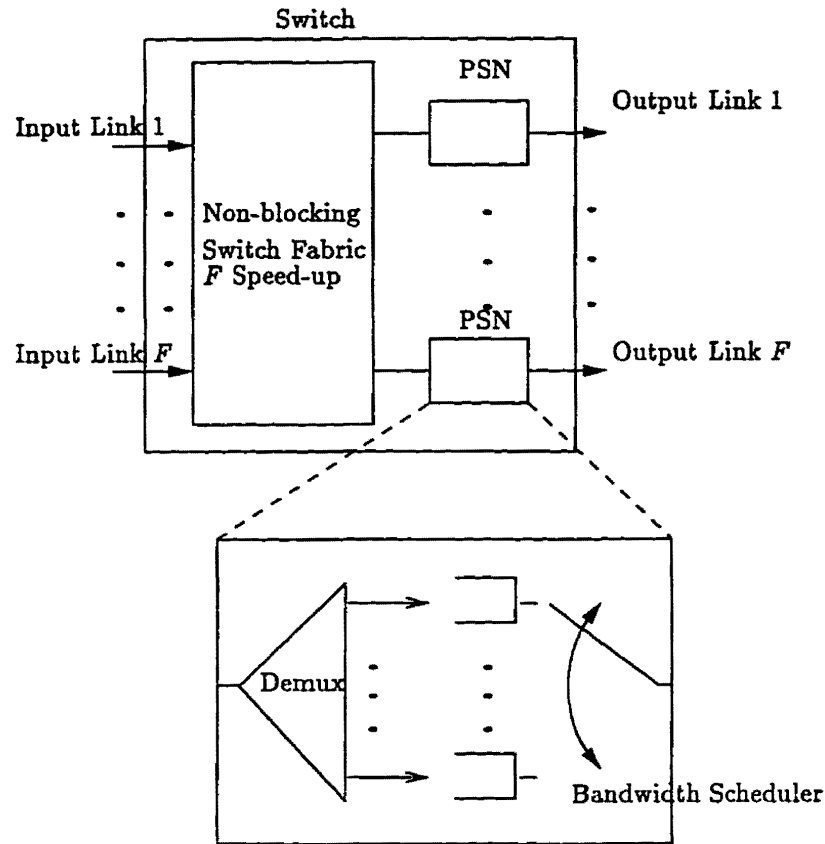


Figure 1.2: Processor sharing node

design: a virtual channel goes through nodes which are the contention points for a fixed output link capacity. This is the assumption in the early parts of this thesis because it simplifies the discussion of queueing by allowing the reader to avoid being distracted by switching issues. Once the queueing issues have been presented, this thesis will show that, with some modifications of the design, PSNs can be used at the input ports of input-buffered switches also.

## 1.1 The Expected Contributions of this Thesis

This thesis presents the design goals for bandwidth scheduling in output-buffered switches and input-buffered switches. As will be shown, the ability to guarantee service bandwidth is the most important design goal. Although the term bandwidth can be used in many different contexts to mean slightly different things, this thesis will give a precise definition of bandwidth guarantees and a complete view of the various aspects of bandwidth guarantees. All the work is done using deterministic (non-probabilistic) calculations.

The following is a list of the specific original contributions in this thesis.

1. The *minimum-bandwidth property*, which is a criterion of how well a scheduler can guarantee a given rate of service, is introduced.
2. The minimum-bandwidth property of the Self-Clocked Fair Queueing and Virtual-Clock schedulers are derived.
3. A modified form of a previously described scheduler, Hierarchical Round-Robin, is introduced and its minimum-bandwidth property is derived.
4. The minimum-bandwidth property is compared with a similar criterion called the service curve.
5. The concept of an effective minimum-bandwidth property is defined which is then used to derive the end-to-end delay bound of tandem PSN queues.
6. Assuming that the arrival process to a PSN queue satisfies a given burstiness envelope, buffer size requirements which ensure no overflow are derived.
7. The burstiness envelope of a departure process is shown to be smaller for schedulers which are non-work-conserving.

8. The effective minimum-bandwidth property of a connection that is multiplexed into a virtual path is derived.
9. The problem of transmission of prerecorded video is described and a solution is proposed.
10. The transmission requirements for prerecorded video, i.e., buffer sizes and playback delay, are derived.
11. A discussion of the scheduling implementations is given with the conclusion that Hierarchical Round-Robin is the best scheduler from both the performance and implementation point of view.

## 1.2 The Organization of this Thesis

Chapter 2 reviews the well-known principles used throughout the thesis. The contributions described previously are organized in the following chapters; also mentioned throughout these chapters are the related contemporary results by others in the literature. Chapter 3 discusses the requirements and solutions for a single PSN queue; i.e., the minimum-bandwidth property and related results. Chapter 4 gives results which are extensions of the results of Chapter 3: the end-to-end delay bounds, buffer sizes, etc. Chapter 5 shows how the results can be used for the prerecorded video application. Chapter 6 raises implementation issues; the design of an input-buffered switch, based on [1], is given as the more implementable alternative to the output-buffered switch. Chapter 7 concludes this thesis with a summary and some conclusions. A glossary of the key terms and mathematical symbols used is given in Appendix A.

## Chapter 2

# The Basic Model and Background Material

This section lays out the modeling assumptions and gives some background material which is necessary for a good understanding of the material in this thesis.

### 2.1 Discrete-Time Modeling

The ATM network operates in discrete time. At each switch, there is a clock which ticks at the rate of once every  $c^{-1}$  seconds, where  $c$  cells/sec is the bandwidth or capacity of the output links. Cells arrive and depart in the intervals between the clock ticks. The interval between ticks are called slots. For simplicity, it is assumed that the capacities of the links within the network are the same ( $c$  cells/sec). This assumption is not necessarily a requirement of ATM networks; however, it makes for much simpler analysis and the analysis can easily be extended to the case where this is not true.

In this thesis, all times are assumed to be in units of slots and bandwidth is always in units of cells/slot, unless otherwise stated. All the analytical work is done in the  $\mathbb{R}^+$

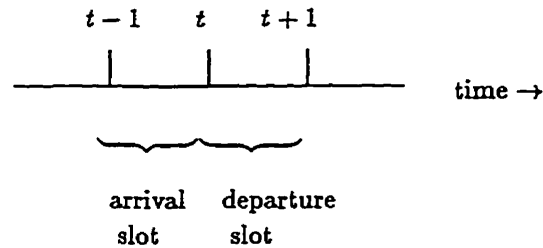


Figure 2.1: Shortest possible queueing delay

domain where  $R^+$  is the set of nonnegative reals. Reals are used despite the fact that the switches operate in discrete time because many values used during analysis may be reals; e.g., bandwidth. To operate in the  $R^+$  domain, the arrival and departure times must be denoted by an instant in the real time line. Arrivals and departures are defined to be when a cell has completely arrived at or departed from a PSN. Since the boundaries of a slot are integers, the arrival and departure times belong to the set  $Z^+$  where  $Z^+ = \{0, 1, 2, \dots\}$ .

Another point to note is that when a cell completely arrives at a PSN, say, at time  $t$ , then the earliest time at which it can begin to depart the PSN is also  $t$  (neglecting switching and processing delays). In this earliest case, the cell completely departs the node at time  $t + 1$ .<sup>1</sup> See Figure 2.1. The delay of one slot is normally referred to as the transmission delay. However, in this thesis, it will be considered to be part of the overall queueing delay. This means that all cells arriving at a PSN will experience a minimum queueing delay of 1 slot.

The points made in this section may seem subtle; however, clarifying them is necessary for a good understanding of much of the analytical work in this thesis.

---

<sup>1</sup>In §3.3.3, the notion of *cut-through* will be introduced in which it is possible to for a cell to depart at the same time that it arrives at the PSN.

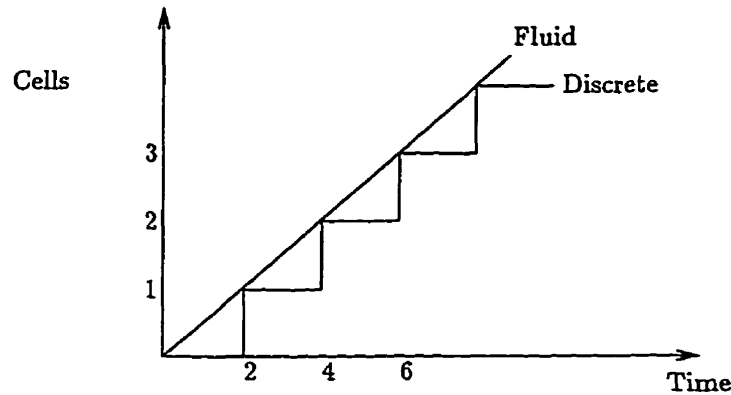


Figure 2.2: Work done in the fluid and discrete-time models

## 2.2 Fluid Modeling

Although the actual physical model of the ATM network is packetized and operates in discrete time, it can be modeled as a *fluid* network. Cells in a fluid network are infinitely divisible and multiple queues can be served at the same time. Consider the following example of how the discrete and fluid models differ. There are two queues in a PSN which are designed to share an output link equally. In the discrete-time model, the sources alternate transmission; i.e., each is on for a cell time, then off, then on again, and so forth. However, in the fluid model, because each source gets exactly half of the link bandwidth, both sources are assumed to be transmitting at a rate of exactly 0.5 cells/slot for the period that the sources are transmitting. Figure 2.2 shows this difference between the actual work done on one of the queues in the discrete-time case and the fluid case, where work is defined as the amount of cells transmitted.

The fluid assumption is good approximation of what really happens in the ATM network because the cells are so small that, at a large time scale, the network behaves virtually like a fluid. The main reason for using fluid models is that they easily allow for

tractable results when compared to an exact discrete representation. A common method used in network analysis is to obtain results based on the fluid assumption, and then derive a bound for the deviation of the actual discrete case to the fluid case. This method is used throughout the thesis; i.e., a discrete-time network model, which uses integers, is analyzed with a continuous-time model, which uses reals.

The use of a fluid model can also incorporate elements of the discrete-time packetized model. In particular, a fluid switch provides service in a fluid manner; however, its arrivals are still discrete packets. In this model, when a packet is said to arrive at time  $s \in \mathbb{Z}^+$ , it means that the cell arrives at the instant  $s$  on the real time line. Note that the cell is eligible to begin service at the instant  $s$ , but when a cell is served at time  $t$ , it means the entire “fluid of the cell” has been served. As an example, consider when a cell arrives at time  $s$ , it can depart the soonest if it is given the entire service bandwidth after  $s$ , in which case its service will be completed at time  $s + 1$  (see Figure 2.1). Note that this is consistent with the packetized model.

### 2.3 The Lindley Process

The discrete-time Lindley process [2] can be used to describe the occupancy of a queue. Let  $A(t)$  and  $D(t)$  be the number of arrivals and departures, respectively, at time  $t$  where  $t \in \mathbb{Z}^+$ . The occupancy of a queue at time  $t$  is

$$Q(t) = Q(t - 1) + A(t) - D(t), \quad (2.1)$$

which is in the form of a Lindley process. Next let  $A(s, t)$  and  $D(s, t)$  be the number of arrivals and departures, respectively, in the interval  $(s, t]$ . It has been shown in [2, page

80] that this can be expressed non-recursively as

$$Q(t) = \max_{0 \leq s \leq t} [A(s, t) - D(s, t)], \quad (2.2)$$

where  $t, s \in Z^+$  and  $A(t, t), D(t, t) = 0$ . Let  $B$  be the maximum queue occupancy, then

$$B = \max_{0 \leq t \leq \infty} Q(t) \quad (2.3)$$

$$= \max_{0 \leq t \leq \infty} \max_{0 \leq s \leq t} [A(s, t) - D(s, t)] \quad (2.4)$$

which implies that

$$B \geq A(s, t) - D(s, t) \quad (2.5)$$

for all  $t > s \geq 0$  where  $t, s \in Z^+$ . Note that the buffer occupancy is largest just after arrivals, which must be integer times by the discrete-time assumption; so, although  $B$  is defined in (2.3) as the maximum occupancy for  $t \in Z^+$ , it is also the maximum occupancy for all  $t \in R^+$ . Therefore, (2.5) holds for all  $t > s \geq 0$  where  $t, s \in R^+$  assuming that arrivals (but not necessarily departures) are in integer times. The significance of (2.5) is that if  $A(s, t)$  and  $D(s, t)$  can somehow be characterized such that a bound  $B$  exists, then the queue can be sized to be  $B$  cells to ensure that overflow never occurs.

## 2.4 ATM Service Classes

The performance requirements of the switches in an ATM network are quite different from those of traditional networks. The traditional networks were generally designed to maximize throughput (packet switching) or to deliver a fixed amount of bandwidth to connections (circuit switching). The ATM switch, however, must be able to deliver a

wide range of services, some of which have stringent QoS requirements, and at the same time give a high overall throughput. The sources can also have highly varying statistical features; consequently, it is desirable to be able to describe the pertinent features of all these sources, as they apply to ATM network design, with a few simple parameters called traffic descriptors. These traffic descriptors are specified in a *contract* that is agreed upon by the user and the network before a connection is established. During the course of the connection, the network ensures that the source/user conforms to these traffic descriptors at the network entry point, called the User Network Interface (UNI), by using a policing device called the Usage Parameter Control (UPC).

There are five well accepted QoS classes of ATM connections [58] [39] [19]. Reference to these service classes will be made throughout this thesis. The Constant Bit Rate (CBR) service can emulate fixed-bandwidth allotment circuit-switching. It has a Peak Cell Rate (PCR) traffic descriptor which is the bandwidth allocation of the connection. The Variable Bit Rate (VBR) service allows for bursty, delay-sensitive traffic such as real-time compressed video. There is also a non-real-time version of VBR; however, this thesis will not emphasize this variety. There are three traffic descriptors most commonly associated with VBR service: PCR, Sustainable Cell Rate (SCR) and Maximum Burst Size (MBS). VBR service also allows for the sharing of network resources to take advantage of *statistical multiplexing* resource utilization gains; how this is to be done requires further study [58]. Unspecified Bit Rate (UBR) service has no QoS guarantees, and is intended only as a “best-effort” service. Best-effort is simply service that the network provides, but which is not guaranteed. Best-effort traffic would receive the resources remaining after the non-best-effort (e.g. CBR and VBR) traffic have used their share.

The remaining two services are less emphasized in this thesis. The Available Bit Rate (ABR) service [6] being proposed has at least the following traffic descriptors: PCR and Minimum Cell Rate ( $MCR \geq 0$ ). Based on availability of network resources, a network

flow control policy dynamically assigns each ABR connection an Allowed Cell Rate (ACR) satisfying  $MCR \leq ACR \leq PCR$ . The ABR class is intended to support non-real-time communications such as data, and there are no delay requirements under ABR service. The bandwidth that ABR uses beyond the MCR (i.e.,  $ACR - MCR$ ) is best-effort, the scheduling aspect of this still requires further study [58]. In addition, specification and research into ABR is still ongoing. The ATM Block Transfer (ABT) service is designed to use a fast reservation protocol [7] to dynamically allocate resources in a block by block basis.

## 2.5 Usage Parameter Control and the $(\sigma, \rho)$ Constraint

The UPC is a “policing” device which imposes a deterministic or statistical constraint on an arrival (or departure) process. The most common policing device discussed in the literature is the Generic Cell Rate Algorithm (GCRA) or “leaky bucket”. Cells arrive at a queue at the leaky bucket, and the leaky bucket determines when the queued cells can be transmitted on the output link in the following manner. The leaky bucket has a token buffer of size  $\sigma$  which is continuously being filled at a *fluid* rate of  $\rho$  tokens/slot. See Figure 2.3. If the token buffer is full, then the newly arriving fluid is lost (due to overflow). A cell in the leaky bucket queue is transmitted whenever there is more than one token worth of fluid in the token buffer. After the cell is transmitted, the size of the token buffer is reduced by one. CBR, VBR and ABR sources can be policed by the leaky bucket so that they obey their traffic descriptors. A leaky bucket essentially imposes a deterministic burstiness constraint called the  $(\sigma, \rho)$  constraint [10].

**Definition 2.1** *An arrival process is said to satisfy a  $(\sigma, \rho)$  constraint if and only if*

$$(t - s)\rho + \sigma \geq A(s, t) \quad (2.6)$$

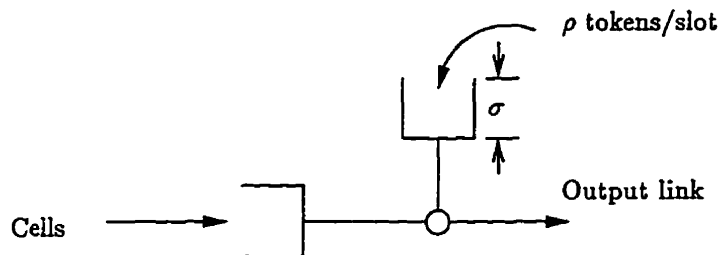


Figure 2.3: A leaky bucket

for all  $t > s \geq 0$ , and  $\sigma, \rho > 0$ .

The definition of (2.6) is equivalent to stating that an arrival source is  $(\sigma, \rho)$ -constrained if and only if, when it arrives at a queue that serves at a fluid rate of exactly  $\rho$  cells/slot, the queue occupancy will not exceed  $\sigma$  cells. See Figure 2.4.<sup>2</sup> The value  $\sigma$  can be considered the *burstiness* of the process which has a long-term average rate of at most  $\rho$ . Note that it is always assumed  $\sigma \geq 1$ , since in a discrete-time model, cells are always served completely at an instant (the departure epoch). What the leaky bucket does can also be referred to as *traffic shaping*. The  $(\sigma, \rho)$  constraint will be used throughout the thesis and, as will be shown in later chapters, it is possible to achieve lossless transmission (assuming no transmission or processing errors) through a PSN with such a constraint.

Imposing a  $(\sigma, \rho)$  constraint means that deterministic results can be obtained between the UNIs of the source and destination end-users. However, sources can still be probabilistic which would mean that there is a probability of buffer overflow. In a virtual channel with a leaky bucket UPC, all the buffer overflow would occur at the leaky bucket. So, when the network is using a leaky bucket, it is essentially moving the problem of statis-

---

<sup>2</sup>The  $\sigma$  of Figure 2.4 and Figure 2.3 are assumed to be the same; this is totally accurate if the network does not operate in discrete time. However, discretizing means that there may be a slight jitter when rounding times off to the nearest integer, which would cause the two  $\sigma$ 's to be slightly different. This slight difference is ignored in this thesis.

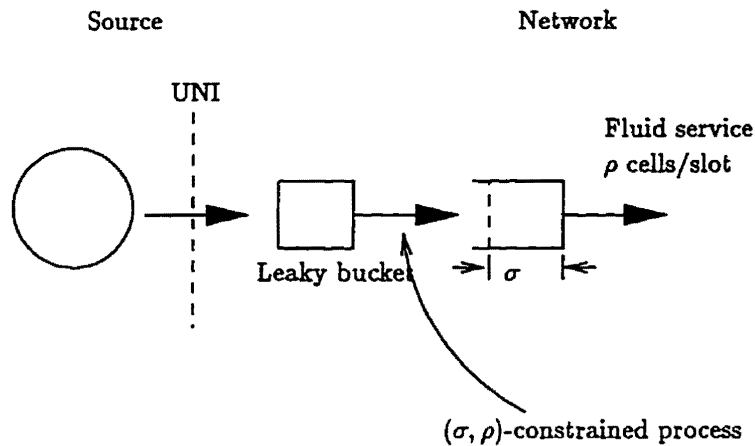


Figure 2.4: Lossless transmission for a  $(\sigma, \rho)$ -constrained arrival process

tical burstiness from the network to the network edges. In other words, it is forcing the user to deal with this problem by, for example, adjusting its rate. Application vendors are working towards interfaces which can transmit arrival processes to the network in a manner that complies with the VBR or CBR traffic descriptors (i.e., the  $(\sigma, \rho)$  constraint). For example, standards have been proposed for using AAL1, which is a CBR service, to transmit MPEG video [41]; to do so, the MPEG encoding scheme would have to alter its compression parameters so that the encoder output is roughly CBR.



## Chapter 3

# The Minimum-Bandwidth Property of Schedulers

### 3.1 PSN Design Goals

A bandwidth scheduling scheme, as shown in Figure 1.2, should be chosen taking into consideration the ATM requirements given in §2.4.

A notable component in the definition of the ATM traffic classes is that CBR, VBR, ABR and, possibly, ABT all offer some kind of sustainable or guaranteed rate of service; i.e., the PCR of CBR, SCR of VBR, and MCR of ABR. Such requirements mean that schedulers must always protect each connection from interference by other connections so that it is never deprived of this guaranteed rate of service. In fact, the network can simply provide this bandwidth in a CBR manner, which is quite implementable; for example, by using round-robin methods (c.f. §3.3.3). In addition, CBR and VBR are fundamental services that ATM must provide; they make possible voice and video conversations, transmission of live events, transmission of prerecorded video for immediate playback, “trunking” services, to name a few. All this must be done in a packet switching

environment where there is also best-effort traffic.

So, the ability to guarantee bandwidth should be the first requirement of a scheduler. The second requirement is that the scheduler must be implementable at ATM speeds — in the hundreds of Mbps at least — and at a reasonable cost. The third requirement is that the scheduler must perform its functions while utilizing network resources efficiently, especially bandwidth, so as to allow more users to use the network. The fourth and last goal, which is less important than the first three, is how the “idle bandwidth”, made available when a queue with guaranteed bandwidth is empty, is distributed by the scheduler. The idle bandwidth can be used to do two things to increase revenue to the network provider. The first is to improve QoS of non-best-effort connections. This improves the “confidence” of network’s statistical guarantees. As an example, consider a VBR connection that has some SCR which was estimated off-line to be able to guarantee a certain cell loss ratio. Such estimates would have a certain confidence interval, meaning that the connection, when actually transmitting, may not get its QoS requirement over its lifetime. For this reason, the idle bandwidth can be used to increase the QoS, and hence the confidence or certainty that the QoS requirements can be satisfied. This helps to keep the customers satisfied with the services of the network. The second use for the idle bandwidth is to increase throughput to best-effort traffic. This may be especially useful if the network charges by the amount of data sent.

The cells of the connection reside at a queue at the PSN, as in Figure 1.2. Connections can be switched alone as a VCC, or they can be bundled together as a Virtual Path Connection (VPC) and be switched that way. In this thesis, it is assumed that all cells of connections with bandwidth guarantees are queued on a per-VCC or per-VPC basis. So, if a connection is part of a VPC, it shares the queue with the other connections in the VPC, otherwise, it has its own queue. A connection with a guaranteed bandwidth requirement must specify its bandwidth requirement during the connection establishment

phase. Let  $N$  be the number of queues with bandwidth guarantees in a PSN. Suppose that a new queue, queue  $N + 1$ , requires  $\rho_{N+1}^{\text{req}}$  cells/slot as a result of a new connection establishment attempt in a new virtual channel. The new connection is accepted at the PSN only if the scheduler can allocate to the queue a bandwidth  $\rho_{N+1} \geq \rho_{N+1}^{\text{req}}$  where the total excess (unallocated) bandwidth at the PSN is

$$U = 1 - \sum_{j=1}^N \rho_j \geq \rho_{N+1} \quad \text{cells/slot.} \quad (3.1)$$

When a queue is allocated a rate of service that is less than its bandwidth requirement; i.e., when  $\rho_{N+1} < \rho_{N+1}^{\text{req}}$ , it is called *overbooking*. Clearly, it is not possible to guarantee a queue a rate of service that is less than its bandwidth requirement; so, *overbooking* cannot satisfy the required QoS. However, the scheduler may be inefficient in a manner such that it is necessary to allocate  $\rho_{N+1} > \rho_{N+1}^{\text{req}}$ , which is called *underbooking*. Underbooking is due to inefficient resource allocation and the difference or underbooked value  $(\rho_{N+1} - \rho_{N+1}^{\text{req}})$  reflects how inefficient a scheduler is. Significant underbooking is not recommended since efficient use of bandwidth is one of the design goals. However, some schedulers may underbook due to a *bandwidth granularity* problem — see the round-robin schedulers in §3.3.3 — which makes it not possible to allocate exactly  $\rho_{N+1}$  bandwidth. In this case, the underbooked value is not too large, and hence the slight underbooking can be out-weighted by other factors such as implementability.

It should be noted here that there is a clear distinction between excess and idle bandwidth. Excess bandwidth is bandwidth made available at the connection level, and it is a long term resource compared with idle bandwidth. Idle bandwidth is actually the sum of the service slots that are made available when they are unused by the queues that they are intended for because the queues are empty. Idle bandwidth occurs at a small time scale, and thus the scheduler cannot take advantage by allocating the idle

bandwidth.

As mentioned earlier, the allocated bandwidth can be the PCR of CBR, SCR of VBR or MCR of ABR. The ABR's ACR rate above the MCR can also be considered a bandwidth guarantee that is provided in a best-effort manner. Viewing the ACR this way means that the idle bandwidth  $U$  should be temporarily allocated to ABR traffic whenever possible, and it must be deallocated whenever a non-best-effort connection (e.g., CBR or VBR) needs the bandwidth. The ABT class may be treated in the same manner. Note that this is only one way to treat ABR and other methods are also possible; for example, by using statistical means to depend on having sufficient available idle bandwidth to service ABR traffic (as opposed to explicitly allocating bandwidth).

Although the main concern is in providing service guarantees to connections that require it, schedulers should also have a method of distributing the idle bandwidth  $U$  to UBR traffic. This can be done by aggregating all UBR connections into a single logical queue. This logical queue could also be temporarily allocated the remaining excess bandwidth after ABR connections have used it. Each time that this logical queue gets service, a secondary scheduler may be used to schedule service to the UBR connections. It will be assumed that all schedulers mentioned in this thesis use this method.

## 3.2 The Minimum-Bandwidth Property

Given that sufficient bandwidth has been allocated, a criterion is presented here which ensures that a particular queue  $n$  is guaranteed a minimum rate of service  $\rho_n$  by the scheduler, called the minimum-bandwidth property. This minimum-bandwidth property is the definition (from the point of view of this thesis) of what is meant by a bandwidth guarantee.

Let the cells that arrive at queue<sup>1</sup>  $n$  be denoted by  $\{c_i^n \mid i \in \{1, 2, \dots\}\}$ . Their arrival times are  $\{a_i^n \mid i \in \{1, 2, \dots\}\}$ . Now suppose that these arrivals are sent to a *fluid* queue which is given a service rate of *exactly*  $\rho_n$  cells/slot. Then, the departure times from the queue are at intervals of exactly  $\rho_n^{-1}$  slots during busy periods — a busy period is an interval when a queue is continuously nonempty — this applies even if  $\rho_n^{-1}$  is not an integer. To define this more precisely, let  $\mathcal{F}_i^n$  be the departure time of cell  $c_i^n$  from this fluid queue. The departure time can be defined recursively as follows:

$$\begin{aligned}\mathcal{F}_i^n &= \max\{\mathcal{F}_{i-1}^n, a_i^n\} + \frac{1}{\rho_n}, \\ \mathcal{F}_0^n &= 0.\end{aligned}\tag{3.2}$$

The  $\mathcal{F}_i^n$ 's are called VirtualClock-Virtual Finishing Times (VC-VFTs) based on  $\{a_i^n\}$  and  $\rho_n$ . Note that only a fluid queue can always offer service to every queue at exactly every  $\rho_n^{-1}$  slots during a busy period since  $\rho_n^{-1}$  can be a fraction. The fluid queue and the VC-VFTs will form the reference for a scheduler which must guarantee a given rate of service.

Now consider again the actual PSN queue; let  $\{d_i^n \mid i \in \{1, 2, \dots\}\}$  be the departure times of cells from this queue.

**Definition 3.1** *A bandwidth scheduler is said to have a minimum-bandwidth property of  $\mu$  slots, where  $\mu \in [1 - \rho_n^{-1}, \infty)$ , if and only if*

$$d_i^n \leq \mathcal{F}_i^n + \mu,\tag{3.3}$$

for all arrival processes, all  $n \in \{1, 2, \dots, N\}$  and all  $i \in \{1, 2, \dots\}$ .

---

<sup>1</sup>When referring to a queue, it is assumed that the queue is a queue with a bandwidth guarantee, unless otherwise stated.

Figure 3.1 gives a graphical presentation of the model for the minimum-bandwidth property. In order to simplify matters, it is assumed that there is never cell loss due to overflow; i.e., the queue size is taken to be infinite in this chapter. In the event that there is cell loss due to overflow, the model can be modified by considering that the lost cells do arrive at the reference queue but of course not at the PSN queue. However, the lost cells do not have departure times and so (3.3) does not apply to lost cells; it only applies to those cells that are not lost. The significance of (3.3) is that  $\mu$  is a bound on how much the queueing delay in a given queue can be greater than the delay through a reference queue which serves at the rate of exactly  $\rho_n$ . Note that the minimum-bandwidth property must hold for all arrival processes, including those of infinite duration. The minimum-bandwidth property may be a function of the bandwidth allocation  $\rho_n$  for a particular queue  $n$ . If  $\mu = \infty$  is the only value that can satisfy (3.3) for a particular scheduler, then that scheduler does not have a minimum-bandwidth property. The minimum-bandwidth property has a lower bound of  $1 - \rho_n^{-1}$ , meaning it could be negative; this is due to the fact that  $d_i^n \geq a_i^n + 1$  and (3.3); i.e.,

$$a_i^n + 1 \leq \mathcal{F}_i^n + \mu \quad (3.4)$$

$$\leq \max\{\mathcal{F}_{i-1}^n, a_i^n\} + \rho_n^{-1} + \mu \quad (3.5)$$

$$\Rightarrow a_i^n - \max\{\mathcal{F}_{i-1}^n, a_i^n\} + 1 - \rho_n^{-1} \leq \mu, \quad (3.6)$$

and since the maximum value of  $a_i^n - \max\{\mathcal{F}_{i-1}^n, a_i^n\}$  is 0, the smallest  $\mu$  is  $1 - \rho_n^{-1}$ . This formulation of the minimum-bandwidth property was independent of that proposed in [25].

A significant feature of the minimum-bandwidth property is that it is an *if and only if* condition on whether the scheduler can guarantee a given rate of service; i.e., if a scheduler does not have a minimum-bandwidth property, then it cannot guarantee a

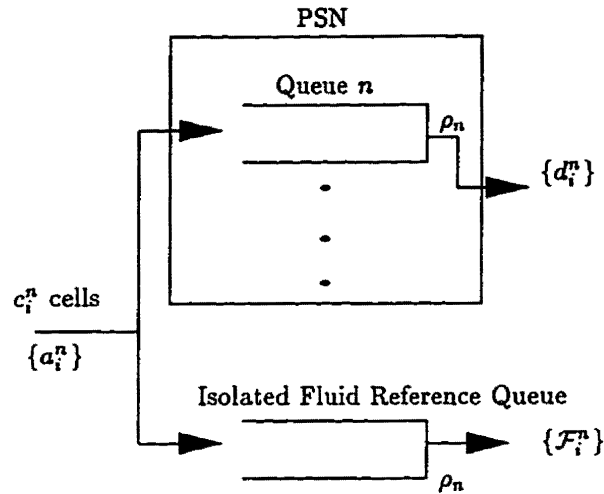


Figure 3.1: Coupling PSN queue to the reference queue

given rate of service. So, the minimum-bandwidth property tells something very critical about schedulers: whether it can guarantee a given rate of service and, if so, how well it does this. §4.2 will define rigorously what this means in the context of the end-to-end delay performance of  $(\sigma, \rho)$ -constrained sources.

### 3.3 Schedulers with Minimum-Bandwidth Properties

There are generally two ways to provide bandwidth guarantees among the proposed bandwidth schedulers. One method is to use a numerical service deadline (a timestamp) for each cell and use that to determine which cell to serve, usually in order of smallest to largest timestamp with ties broken arbitrarily. The other is to service the queue in some round-robin (cyclic) fashion using frames. This section looks at the minimum-bandwidth properties of some of the well-known schedulers in the literature. A taxonomy of all well-known schedulers is given in [67].

### 3.3.1 Work-Conserving VFT Schedulers

A common class of timestamping schedulers in the literature uses *virtual finishing times* (VFTs). When a cell  $c_i^n$  arrives at the node at a time  $a_i^n$ , the scheduler timestamps it with a VFT, denoted as  $F_i^n$ . For all  $n \in \{1, 2, \dots, N\}$  and  $i \in \{1, 2, \dots\}$ , the VFTs are determined recursively as follows:

$$\begin{aligned} F_i^n &= \max\{F_{i-1}^n, v(a_i^n)\} + \frac{1}{\rho_n}, \\ F_0^n &= 0 \end{aligned} \quad (3.7)$$

where  $v(\cdot)$  is the *virtual time function* of the particular scheduler. All the VFT schedulers described in this section chooses the cell with the smallest VFT at each slot. In particular, the cell chosen to depart at time  $t$  is the least VFT in the PSN after all the cells have arrived and departed at time  $t-1$ .<sup>2</sup> Cells that have the same VFT are chosen in arbitrary order.

The first VFT scheduler in this section is the Packetized Generalized Processor Sharing (PGPS) scheduler [54] [15]. The goal of this scheduler is to approximate an idealized fluid model of data flow through the switch, called Generalized Processor Sharing (GPS). In this fluid model, if queue  $n$  is nonempty at (continuous) time  $t$ , then the amount of bandwidth it receives is at time  $t$  is *exactly*  $\rho_n / \sum_{i \in A(t)} \rho_i$ , where  $A(t)$  is the index set of nonempty queues at  $t$ . In other words, a nonempty queue in the fluid model would receive service in proportion to all the other nonempty queues. In this thesis this will be referred to as *GPS fairness*. It has been shown in [54] [27] that GPS has a minimum-bandwidth property of  $\mu = 0$ , but GPS is not implementable because it is a fluid scheduler. However,

---

<sup>2</sup>Another view of this is that just after the arrivals and departures at time  $t-1$ , the work-conserving scheduler chooses the next cell to serve based on the current VFTs. This cell is served in the interval  $(t-1, t]$ . However, because a cell's departure time is when it has completely departed from the PSN, the departure time of the chosen cell is at time  $t$ .

theorems exist in [54] [27] which show that PGPS “tracks” GPS. This property, combined with the fact that GPS has a minimum-bandwidth property of  $\mu = 0$ , implies that PGPS has a minimum-bandwidth property of  $\mu = 0$  also.

### VirtualClock

A very well known VFT scheduler is VirtualClock [68]. This scheduler is characterized by simply substituting  $v(t) = t$  into (3.7). The VFTs in this case are called VirtualClock-Virtual Finishing Times which are identical to the  $\mathcal{F}_i^n$ 's in (3.2) based on  $\{a_i^n\}$  and  $\rho_n$ . L. Zhang first proposed this scheduler because she considered the VFTs to be the times at which cells are served if the scheduler uses a Time Division Multiplexing (TDM) scheme. However, there is an intuitive difference between VirtualClock and TDM: with TDM, if a queue does not have cells to send, the slot is not utilized to serve the other queues (non-work-conserving), with VirtualClock, slots are utilized whenever there is a nonempty queue (work-conserving). It is obvious then that VirtualClock can give more bandwidth to a queue than what is allocated, through the idle bandwidth.

The following proves the minimum-bandwidth property for the VirtualClock scheduler; the proof was inspired by [54]. This result was arrived at independently from other contemporary work [65] [16].

**Theorem 3.1** *The VirtualClock scheduler has a minimum-bandwidth property of  $\mu = 0$ .*

*Proof:* Consider an arbitrary cell  $c_i^n$ . Let  $H \in \mathbb{Z}^+$  be the largest integer such that at time  $H$ , where  $H \leq \mathcal{F}_i^n$ , the PSN is idle (i.e., no cell departs the PSN at time  $H$  and a cell departs the PSN at every slot in the interval  $[(1 + H), \mathcal{F}_i^n]$ ). If no such  $H \geq 0$  exists, take  $H = 0$ .

Let  $J \in \mathbb{Z}^+$  be the largest integer such that at time  $J$ , where  $J \leq \mathcal{F}_i^n$ , a cell departs the PSN with VFT  $> \mathcal{F}_i^n$ . If no such  $J \geq 0$  exists, take  $J = 0$  as well.

To prove the theorem, first consider the case where  $a_i^n < (\max\{H, J\} - 1)$ . In this case,  $d_i^n < \max\{H, J\} \leq \mathcal{F}_i^n$ , since  $c_i^n$  cannot be queued at time  $\max\{H, J\}$  due to the definition of  $H$  and  $J$ . In addition,  $a_i^n \neq (H - 1)$  and  $a_i^n \neq (J - 1)$  because  $c_i^n$  cannot be queued at time  $\max\{H, J\}$ .

For the case where  $a_i^n > (\max\{H, J\} - 1)$ , consider the intervals of time  $I = [\max\{H, J\}, \mathcal{F}_i^n]$  and  $\hat{I} = [(1 + \max\{H, J\}), \mathcal{F}_i^n]$ . The following shows that every cell  $\zeta$  which has  $\text{VFT} \leq \mathcal{F}_i^n$  and arrival time  $a \in I$  will depart the PSN during the interval  $\hat{I}$ . Clearly, the VFT of  $\zeta$  is  $\leq \mathcal{F}_i^n$  by the definition of  $J$ . Note that the length of interval  $I$  is  $|I| = \mathcal{F}_i^n - \max\{H, J\}$ . The number of cells  $\zeta$  with  $\text{VFT} \leq \mathcal{F}_i^n$  and arrival time  $a \in I$  is

$$\leq \left\lfloor \sum_{i=1}^N \rho_i |I| \right\rfloor \leq \lfloor |I| \rfloor;$$

one of these cells is  $c_i^n$ . Since exactly  $\lfloor |I| \rfloor$  cells depart the PSN in the interval  $\hat{I}$  — note that a cell departs at time  $(1 + \max\{H, J\})$  —  $c_i^n$  must depart in  $\hat{I}$ .

To summarize, it has been shown that if  $a_i^n < (\max\{H, J\} - 1)$  then  $d_i^n < \max\{H, J\}$ ; and if  $a_i^n \in I$  then  $d_i^n \in \hat{I}$ .

□

### Self-Clocked Fair Queueing

Although PGPS has GPS fairness properties, the virtual time function is more difficult to implement than VirtualClock. To address this, a much more feasible VFT scheduler has been proposed [24] [57]. With this scheduler  $v(t)$  is the VFT of the last cell to depart the PSN at or before  $t$ ; if no cell has departed the PSN at or before time  $t$ ,  $v(t) = 0$ .<sup>3</sup>

<sup>3</sup>Mathematically, the virtual time function can be expressed as

$$v(t) = \begin{cases} 0 & \text{if } \{(n, i) | d_i^n \leq t\} = \emptyset \text{ else} \\ F_i^n & \text{where } (n, i) = \arg \sup \{d_i^n | d_i^n \leq t\} \end{cases}$$

This virtual time function, like VirtualClock, is very simple to implement. It was argued that this approach, called Self-Clocked Fair Queueing (SCFQ), can approximately provide GPS fairness [24] [57].

**Theorem 3.2** *The SCFQ scheduler has a minimum-bandwidth property of  $\mu = N - 1$ .*

*Proof:* For an arbitrary cell  $c_i^n$ , let  $k \leq i$  be the largest integer such that  $d_{k-1}^n < a_k^n$ ; if no such  $k$  exists set  $k = 1$ . This means that  $c_k^n$  is the last cell to arrive at an empty queue  $n$  before or at time  $a_i^n$ .

The following shows that, if  $k < i$ ,

$$F_j^n = F_{j-1}^n + \frac{1}{\rho_n} \quad \forall j \in \{k+1, k+2, \dots, i\}. \quad (3.8)$$

The definition of  $k$  implies that  $c_j^n$  arrives to a nonempty queue (i.e.,  $c_{j-1}^n$  is queued at time  $a_j^n$ ). Since  $v$  is nondecreasing (proved by Lemma 2 of [24]),  $v(a_j^n) \leq F_{j-1}^n$ . Therefore, (3.8) follows directly from (3.7).

The following shows that

$$F_k^n = v(a_k^n) + \frac{1}{\rho_n} \quad (3.9)$$

This is clearly true for  $k = 1$ . For  $k > 1$ , note that  $c_{k-1}^n$  is served before  $a_k^n$ . So, since  $v$  is nondecreasing,  $v(a_k^n) \geq F_{k-1}^n$ . (3.9) then follows directly from (3.7).

By (3.8) and (3.9),

$$F_i^n = v(a_k^n) + \frac{i - k + 1}{\rho_n}. \quad (3.10)$$


---

Because  $c_i^n$  arrives during a busy period initiated by  $c_k^n$ , precisely  $i - k + 1$  cells depart SCFQ queue  $n$  over  $[a_k^n, d_i^n]$ . Therefore,

$$d_i^n \leq a_k^n + \left( i - k + 1 + \sum_{m \neq n} \gamma_m \right) \quad (3.11)$$

where  $\gamma_m$  is the maximum number of queue  $m$  cells that can depart the SCFQ node in  $[a_k^n, d_i^n]$ ; all such cells must have VFTs less than or equal to  $F_i^n$  and greater than or equal to  $v(a_k^n)$ .

Since  $F_{j+1}^m - F_j^m \geq \rho_m^{-1}$  for all  $j \geq 1$  and  $m \in \{1, 2, \dots, N\}$ ,

$$v(a_k^n) + \frac{\gamma_m - 1}{\rho_m} \leq F_i^n = v(a_k^n) + \frac{i - k + 1}{\rho_n}. \quad (3.12)$$

where the second inequality is (3.10). The  $v(a_k^n)$  terms in this inequality cancel out giving  $\gamma_m \leq 1 + (i - k + 1)\rho_m/\rho_n$ . Substituting this into (3.11) gives

$$d_i^n \leq a_k^n + \left( i - k + 1 + \sum_{m \neq n} \left( 1 + (i - k + 1) \frac{\rho_m}{\rho_n} \right) \right) \quad (3.13)$$

$$\leq a_k^n + \left( i - k + 1 + N - 1 + (i - k + 1) \frac{1 - \rho_n}{\rho_n} \right) \quad (3.14)$$

$$= a_k^n + \left( \frac{i - k + 1}{\rho_n} + N - 1 \right). \quad (3.15)$$

Clearly, for all  $j \in \{1, 2, \dots, i\}$ ,

$$\mathcal{F}_i^n \geq a_j^n + \left( \frac{i - j + 1}{\rho_n} \right). \quad (3.16)$$

Subtracting (3.16) with  $j = k$  from (3.15) then gives

$$d_i^n \leq \mathcal{F}_i^n + N - 1. \quad (3.17)$$

□

So, the minimum-bandwidth property is in the order of the number of queues with bandwidth guarantees in the PSN, or  $N$ . The theorem does not show whether this bound is tight, however. The following example addresses this by showing that the minimum-bandwidth property of order  $N$  can be achieved. This worst case is as shown in Table 3.1. There are  $N$  queues, all with the same bandwidth allocation of  $\rho = N^{-1}$ . The first column is the time in units of slots. The last column shows which queue is served at each slot (the number without parenthesis); the value in the parenthesis denotes the VFT of the cell being served. The remaining columns in the middle show which queues experience an arrival and, if so, what the VFT of the arriving cell is. What the table shows is that all queues experience an arrival during the first two slots except queue  $N$  which has an arrival at the second slot only. Then, by following the example, it can be seen that the cell in queue  $N$  is served at time  $2N$ . The cell has a VC-VFT of  $\mathcal{F}_1^N = 2 + N$  by (3.2), thus resulting in a minimum-bandwidth property in the order of  $N$ .

This theorem was proved independently of [25], which derived the same result, and [57], which gives the work done by SCFQ over an arbitrary busy period.

### 3.3.2 Other Timestamping Schedulers

Another scheduler is Start-time Fair Queueing [26]. This scheduler uses VFTs but, unlike the VFT schedulers discussed previously, at each slot it chooses the cell with the least *start-time*; the start-time for  $c_i^n$  is  $S_i^n = F_i^n - \rho_n^{-1}$ . The virtual time function  $v(t)$  is the start-time of the last cell to depart the PSN at or before  $t$ ; if no cell has departed the PSN at or before time  $t$ ,  $v(t) = 0$ . Note that this scheduler is the same as SCFQ, except that this scheduler uses the start-time instead of the VFT as the timestamp to determine what cell to serve and for the virtual time function. In [26] it was shown that this scheduler has a minimum-bandwidth property of  $\mu = N - \rho_n^{-1}$ . Note that  $\mu$  is a value which may be

Time	Queue 1	Queue 2	Queue 3 ... $N - 2$	Queue $N - 1$	Queue $N$	Queue Served
1	$1 + N$	$1 + N$	...	$1 + N$	—	—
2	$1 + 2N$	$1 + 2N$	...	$1 + 2N$	$2 + N$	$1(1 + N)$
3	—	—	...	—	—	$2(1 + N)$
.	—	—	...	—	—	.
.	—	—	...	—	—	.
.	—	—	...	—	—	.
$N$	—	...	—	—	—	$N - 1(1 + N)$
$N + 1$	—	...	—	—	—	$1(1 + 2N)$
$N + 2$	—	...	—	—	—	$2(1 + 2N)$
.	—	—	...	—	—	.
.	—	—	...	—	—	.
.	—	—	...	—	—	.
$2N - 1$	—	...	—	—	—	$N - 1(1 + 2N)$
$2N$	—	...	—	—	—	$N(2 + N)$

Table 3.1: A worst-case scenario for SCFQ

negative for some queues, and this value makes the end-to-end delay independent of the allocated bandwidth  $\rho_n$  (c.f. §4.2). Note also that this is the first scheduler mentioned for which the value of the minimum-bandwidth property differs from one queue to another queue because it is a function of  $\rho_n$ .  $WF^2Q$  [5] and  $WF^2Q+$  [4] are two schedulers that use a similar notion of a start-time; they both have minimum-bandwidth properties of  $\mu = 0$ .  $WF^2Q+$  was shown to have good statistical delay properties.

Idling VirtualClock [33] is a non-work-conserving version of the VirtualClock scheduler proposed by G. Kesidis. The idling VirtualClock uses the same timestamps as VirtualClock but the cell chosen at any time  $t$  is the least VFT  $F_i^n$  such that  $\lceil F_i^n \rceil \leq t$ . This scheduler is non-work-conserving because the PSN may be idle or is serving best-effort traffic even though there is a nonempty queue with a bandwidth guarantee in the PSN. This has the advantage of better control over the idle bandwidth (c.f. §3.7). This scheduler was shown to have a minimum-bandwidth property of  $\mu = N - 1$ . The Airport

scheduler [11] is similar to idling VirtualClock, and it has a similar minimum-bandwidth property.

For all the timestamping schedulers discussed thus far, the time of reference is the virtual time, which is a continuously increasing function. Schedulers have been proposed, such as [55], where the time reference is constant, and the timestamp of each cell is continuously changing in value; however, these implementations are less feasible.

### 3.3.3 Round-Robin Schedulers

#### Weighted Round-Robin

Weighted Round-Robin (WRR) is a simple TDM-like bandwidth scheduler. Under WRR, queues are given service opportunities in a round-robin fashion. Each round-robin cycle is called a WRR frame (of cell slots). Let  $f$  be the number of slots in a WRR frame. In one version, called *non-work-conserving* or *idling WRR*, slots in a frame are reserved to queues in each WRR frame. Suppose that a queue has  $k$  slots reserved for it in each WRR frame; hence, this queue has a bandwidth allotment of  $k f^{-1}$  cells/slot. If the queue does not require any of its reserved number of slots, then those slots can be distributed to other queues as idle bandwidth. In this scheme, the slots given to a queue need not be consecutive (contiguous), but this requires that a “free” list of non-assigned slots be maintained. If the slots must be consecutive, there may be a fragmentation problem when connections are terminated which would make it necessary to re-assign (re-shuffle) slots. Re-assigning adds complexity and may also increase the minimum-bandwidth property.

Another version is called *work-conserving* or *non-idling WRR* where each queue is served once in each frame. The scheduler cycles once through each queue where it is granted up to its allocated number of slots or its current queue occupancy, whichever is less. The duration of a frame here may be less than  $f$  due to unused slots. The

main difference between the two types of WRR is that the service of the idling version is regular and constant, while it is less so in the non-idling version. In the non-idling version, queues with bandwidth guarantees would normally get more than their allocated bandwidth through idle bandwidth distribution, like the work-conserving timestamping schedulers such as VirtualClock, PGPS, etc. Here, the attention will be paid to the idling version because it allows the scheduler to have better control over the idle bandwidth.

Note that the *minimum* amount of bandwidth that can be assigned under WRR is  $f^{-1}$  cells/slot; this is the *bandwidth granularity* of WRR. The bandwidth granularity is coarse if  $f$  is small. On the other hand, the minimum-bandwidth property is in the order of  $f$ , as stated by G. Kesidis in [44, §3.2], which follows the work in this thesis; so, a large  $f$  gives a finer bandwidth granularity but there is a minimum-bandwidth property tradeoff. The following section presents a more sophisticated round-robin scheduler with a minimum-bandwidth property much better than that of WRR.

### Hierarchical Round-Robin

Hierarchical Round-Robin (HRR) [40] is a more flexible kind of round-robin scheduling. HRR attempts to allocate bandwidth more “evenly” over a frame than WRR, and it also offers better bandwidth granularity. In HRR, there are  $L \geq 1$  *levels* of frames. For the level  $l$  HRR frame,  $l \in \{1, 2, \dots, L\}$ , let  $f_l$  slots be the size and let  $n_l < f_l$  be the number of slots that are reserved for level  $l + 1$  where  $n_L = 0$ . Figure 3.2 shows an example with  $L = 3$ ,  $\{f_1, f_2, f_3\} = \{14, 6, 12\}$  and  $\{n_1, n_2\} = \{3, 3\}$ . For this example, slots from level 2 (and in turn, level 3) will be served on the 12<sup>th</sup> – 14<sup>th</sup> slots, 26<sup>th</sup> – 28<sup>th</sup> slots, 40<sup>th</sup> – 42<sup>th</sup> slots, etc.

To allow for a greater variety of bandwidth allocations using the same frame structure, a queue is permitted to have reserved slots on more than one frame level in this proposed version of (idling) HRR called *multilevel-assignment* HRR. Note that in the special case

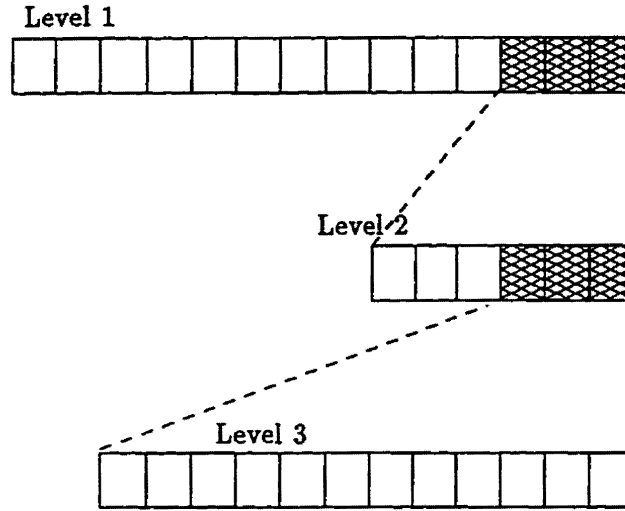


Figure 3.2: An HRR frame structure example

where  $L = 1$ , HRR is just WRR.

This section will consider a particular queue, and the superscript and subscript notation denoting the index of the queue will be implied; i.e.,  $a_i^n$  is just  $a_i$ ,  $\rho_n$  is just  $\rho$ . It can be seen that the particular queue with  $k_l$  slots reserved in the level  $l$  frame has a bandwidth allotment of

$$\rho = \sum_{l=1}^L k_l \frac{n_1 n_2 \dots n_{l-1}}{f_1 f_2 \dots f_{l-1} f_l} \text{ cells/slot.} \quad (3.18)$$

To simplify the subsequent analysis, a simplifying constraint is made:

$$\frac{f_{l+1}}{n_l} \in \{2, 3, 4, \dots\} \quad \text{for all } l \in \{1, 2, \dots, L-1\}. \quad (3.19)$$

This also ensures that the lower levels (i.e., large  $l$ ) have a finer bandwidth granularity

than the higher ones. Let

$$\varphi_l = k_l \frac{n_1 n_2 \dots n_{l-1}}{f_1 f_2 \dots f_l} \quad (3.20)$$

where  $\varphi_1 = k_1/f_1$ ;  $\varphi_l$  is the bandwidth allocated at level  $l$  to the queue under consideration. Thus,  $\rho = \sum_{l=1}^L \varphi_l$ . Also, let  $K_l(s, t)$  be the number of reserved level  $l$  slots to the queue under consideration in the interval  $(s, t]$ .

The following lemmas and theorems will derive the minimum-bandwidth property of this version of idling HRR. Also, for the remainder of this thesis, when referring to HRR, it is assumed to be multilevel-assignment HRR.

First, let  $\xi_l$  be a value such that

$$|K_l(s, t) - (t - s)\varphi_l| \leq \xi_l \quad (3.21)$$

for all  $t > s \geq 0$  where  $s, t \in \mathbb{Z}^+$ ;  $\xi_l$  can be considered to be a bound on the difference in the work done by the allocated HRR slots and a fluid reference queue with rate  $\varphi_l$ . Also, let  $\xi = \sum_{l=1}^L \xi_l$ .

**Lemma 3.1** For all  $l \in \{1, 2, \dots, L\}$ ,

$$\xi_l \leq \frac{k_l}{f_l} (f_l - k_l) + k_l \sum_{i=1}^{l-1} \frac{n_{l-i} \dots n_{l-1}}{f_{l-i} \dots f_l} (f_{l-i} - n_{l-i}). \quad (3.22)$$

*Proof:* Consider an interval of time  $(s, t]$  where  $s, t \in \mathbb{Z}^+$ . Let  $N_l(s, t)$  be the number slots reserved for all level  $l$  queues (and subsequent levels) in the interval  $(s, t]$ . The following first proves the lower bound; i.e.,  $K_l(s, t) - \varphi_l(t - s) \geq -\xi_l$ .

Since there are exactly  $n_{l-1}$  reserved slots for levels  $l > 1$  in every  $f_{l-1}$  level  $l - 1$

slots,

$$N_l(s, t) - \frac{n_{l-1}}{f_{l-1}} N_{l-1}(s, t) = N_l(s, r) - \frac{n_{l-1}}{f_{l-1}} N_{l-1}(s, r) \quad (3.23)$$

where  $r$  is the smallest integer such that  $r \geq s$  and  $N_{l-1}(r, t) \bmod f_{l-1} = 0$  (note  $N_l(t, t) = 0$  for all  $t$  and  $l$ ). In addition,

$$N_l(s, r) - \frac{n_{l-1}}{f_{l-1}} N_{l-1}(s, r) \geq 0 - \frac{n_{l-1}}{f_{l-1}} (f_{l-1} - n_{l-1}); \quad (3.24)$$

i.e., a lower bound is achieved when  $N_{l-1}(s, r) = f_{l-1} - n_{l-1}$  and  $N_l(s, r) = 0$ . Therefore,

$$N_l(s, t) - \frac{n_{l-1}}{f_{l-1}} N_{l-1}(s, t) \geq -\frac{n_{l-1}}{f_{l-1}} (f_{l-1} - n_{l-1}). \quad (3.25)$$

A similar argument can be used to derive a bound on the number of slots reserved to the queue at level  $l$ ; i.e.,

$$K_l(s, t) - \frac{k_l}{f_l} N_l(s, t) \geq 0 - \frac{k_l}{f_l} (f_l - k_l). \quad (3.26)$$

Combining this with (3.25) gives

$$K_l(s, t) - \frac{k_l n_{l-1}}{f_{l-1} f_l} N_{l-1}(s, t) \geq -\frac{k_l}{f_l} (f_l - k_l) - \frac{k_l n_{l-1}}{f_{l-1} f_l} (f_{l-1} - n_{l-1}). \quad (3.27)$$

This method repeated until the fact that  $N_1(s, t) = t - s$  is used to get

$$K_l(s, t) - \varphi_l(t - s) \geq -\frac{k_l}{f_l} (f_l - k_l) - k_l \sum_{i=1}^{l-1} \frac{n_{l-i} \cdots n_{l-1}}{f_{l-i} \cdots f_l} (f_{l-i} - n_{l-i}) \quad (3.28)$$

which gives the lower bound of  $K_l(s, t) - \varphi_l(t - s) \geq -\xi_l$ .

The upper bound of  $K_l(s, t) - \varphi_l(t - s) \leq \xi_l$  can be derived in a very similar manner,

but instead of (3.24), the following is used:

$$N_l(s, \tau) - \frac{n_{l-1}}{f_{l-1}} N_{l-1}(s, \tau) \leq n_{l-1} - \frac{n_{l-1}}{f_{l-1}} n_{l-1} = \frac{n_{l-1}}{f_{l-1}} (f_{l-1} - n_{l-1}); \quad (3.29)$$

i.e., the upper bound is achieved when  $N_{l-1}(s, \tau) = n_{l-1}$  and  $N_l(s, \tau) = n_{l-1}$ . Similarly, instead of (3.26), the following is used:

$$K_l(s, t) - \frac{k_l}{f_l} N_l(s, t) \leq k_l - \frac{k_l}{f_l} k_l = \frac{k_l}{f_l} (f_l - k_l). \quad (3.30)$$

□

Let  $d_i^-$  be the instant just before cell  $i$  departs at  $d_i$ . Numerically, the difference  $d_i - d_i^-$  is assumed to be so small that, essentially,  $d_i = d_i^-$ . Also, let  $Q(t)$  and  $\Omega(t)$  be the queue occupancy of the PSN queue and the fluid queue, respectively, at time  $t$ .<sup>4</sup> Finally recall that  $A(s, t)$  is the number of arrivals in the interval  $(s, t]$  where  $s \leq t$ . In the case where  $s > t$ ,  $A(s, t)$  will be defined to be  $-A(t, s)$ .

**Lemma 3.2** For all  $i \in \{1, 2, \dots\}$ ,

$$Q(d_i^-) = \Omega(\mathcal{F}_i) + A(\mathcal{F}_i, d_i) + 1. \quad (3.31)$$

*Proof:* By the definition of  $d_i^-$  and  $d_i$ ,

$$Q(d_i^-) = Q(d_i) + 1. \quad (3.32)$$

Now consider an arbitrary cell which arrives at a queue at time  $a$ . When it departs that queue at a time  $d$ , the queue occupancy must be the number of cell arrivals in the interval

---

<sup>4</sup>A comment regarding notation: normally, variables of the PSN queue are expressed as alphabetical letters; e.g.,  $Q(t)$  for the queue occupancy. Correspondingly, the same value for the fluid reference queue is always expressed in the same manner but in script notation; i.e.,  $\Omega(t)$ .

$(a, d]$ . Using this reasoning gives

$$\Omega(\mathcal{F}_i) = A(a_i, \mathcal{F}_i) \quad (3.33)$$

and

$$Q(d_i) = A(a_i, d_i) = A(a_i, \mathcal{F}_i) + A(\mathcal{F}_i, d_i) \quad (3.34)$$

which, when combined with (3.32), proves the lemma. Note that (3.34) and the lemma holds even when  $\mathcal{F}_i > d_i$ .

□

Before proceeding further, it is first necessary to introduce the concept of *cut-through*. Recall that §2.1 defined a queueing model where a cell must wait at least one slot before being transmitted; i.e.,  $d_i \geq a_i + 1$ . This one slot is to account for the transmission time. If this restriction is omitted from the queueing model, then it is possible for a cell to be transmitted immediately after it arrives. This would produce a very minor difference because it does not change the ordering of cells from the PSN, but it allows every cell to depart one slot earlier.

**Definition 3.2** *Cut-through:* Given the arrival process  $\{a_i\}$  of a queue, a PSN which has cut-through will produce a departure process  $\{d_i - 1\}$ , where  $\{d_i\}$  is the departure process of the queue if there is no cut-through.

The following analysis assumes a cut-through PSN in order to simplify matters. The occupancy at time  $t \in \mathbb{R}^+$  of the cut-through idling HRR queue can be expressed as

$$Q(t) = \max_{0 \leq s \leq t} \{A(s, t) - K(s, t)\}. \quad (3.35)$$

The occupancy of the fluid reference queue can be expressed as

$$\Omega(t) = \max_{0 \leq s \leq t} \{A(s, t) - \rho(t - s)\}. \quad (3.36)$$

**Lemma 3.3** *Given that cut-through idling HRR is used, for all  $t \in \mathbb{R}^+$ ,*

$$Q(t) \leq \Omega(t) + \xi. \quad (3.37)$$

Proof: By (3.21)

$$\xi \geq \sum_{l=1}^L ((t - s)\varphi_l - K_l(s, t)) \quad (3.38)$$

$$\geq (t - s)\rho - K(s, t). \quad (3.39)$$

Rearranging this inequality and adding  $A(s, t)$  to both sides gives

$$\begin{aligned} A(s, t) - K(s, t) &\leq A(s, t) - \rho(t - s) + \xi \\ \Rightarrow \max_{0 \leq s \leq t} \{A(s, t) - K(s, t)\} &\leq \max_{0 \leq s \leq t} \{A(s, t) - \rho(t - s)\} + \sum_{l=1}^L \xi_l, \end{aligned} \quad (3.40)$$

and the desired result follows from (3.40), (3.35) and (3.36).

□

**Theorem 3.3** *The idling HRR scheduler using cut-through has a minimum-bandwidth property of  $\mu = \rho^{-1}(\xi - 1)$ .*

Proof: Consider the  $i^{\text{th}}$  cell to arrive at the HRR queue under consideration. Assume that  $d_i \geq \mathcal{F}_i$ , otherwise the desired result is immediate.

Using Lemma 3.3 and the fact that  $\Omega(d_i^-) = \Omega(d_i)$  since there can be no arrivals in

$(d_i^-, d_i]$ , and so  $\Omega(\cdot)$  is a continuous function,

$$Q(d_i^-) \leq \Omega(d_i^-) + \xi \quad (3.41)$$

$$\leq \Omega(d_i) + \xi. \quad (3.42)$$

Combining (3.42), Lemma 3.2 and the fact that  $\Omega(d_i) = [\Omega(\mathcal{F}_i) + A(\mathcal{F}_i, d_i) - \rho(d_i - \mathcal{F}_i)]^+$  gives

$$\Omega(\mathcal{F}_i) + A(\mathcal{F}_i, d_i) + 1 \leq [\Omega(\mathcal{F}_i) + A(\mathcal{F}_i, d_i) - \rho(d_i - \mathcal{F}_i)]^+ + \xi, \quad (3.43)$$

Note that varying  $A(\mathcal{F}_i, d_i)$  will have no effect on  $d_i$  or  $\mathcal{F}_i$ . Also, note that (3.43) holds for all possible values of  $A(\mathcal{F}_i, d_i)$ . In particular, (3.43) holds when

$$A(\mathcal{F}_i, d_i) \geq -\Omega(\mathcal{F}_i) + \rho(d_i - \mathcal{F}_i). \quad (3.44)$$

In this case, by (3.43),

$$0 \leq -\rho(d_i - \mathcal{F}_i) + \xi - 1. \quad (3.45)$$

□

A simple explanation of this theorem was pointed out by T. Kameda: consider a time  $s$  that begins a busy period  $(s, t]$  of the PSN and reference queue. See Figure 3.3. The curve  $\underline{D}(s, t)$  is the minimum number of departures from the PSN queue in  $(s, t]$ . Based on the diagram, it can be seen that

$$\rho(d_i - \mathcal{F}_i) + 1 \leq \max_t [\rho(t - s) - \underline{D}(s, t)] \leq \xi \quad (3.46)$$

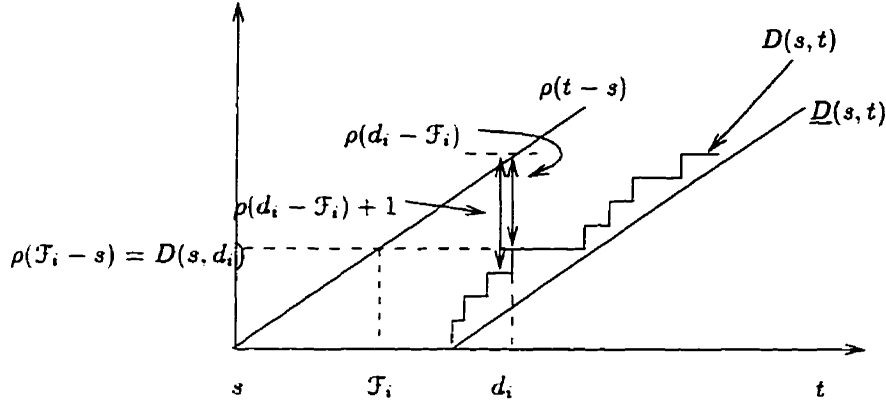


Figure 3.3: Graphical explanation of Theorem 3.3

which is the same as (3.45).

Using Theorem 3.3 and the fact that the departure times from the cut-through queue are one slot earlier than the non-cut-through case — which is what is used throughout this thesis — the following result is immediate.

**Corollary 3.1** *The idling HRR scheduler (without cut-through) has a minimum-bandwidth property of  $\mu = \rho^{-1}(\xi - 1) + 1$ .*

Using Lemma 3.1 gives a minimum-bandwidth property of

$$\mu = \rho^{-1} \sum_{l=1}^L \left( \frac{k_l}{f_l} (f_l - k_l) + k_l \sum_{i=1}^{l-1} \frac{n_{l-i} \cdots n_{l-1}}{f_{l-i} \cdots f_l} (f_{l-i} - n_{l-i}) \right) - \rho^{-1} + 1. \quad (3.47)$$

The minimum-bandwidth property of multilevel-assignment idling HRR can often be better than that of WRR and single-level-assignment idling HRR because the departures are more evenly spaced under multilevel-assignment. For example, consider a HRR frame structure with two levels ( $L = 2$ ) such that  $\{f_1, f_2\} = \{10, 20\}$  and  $n_1 = 2$ . The minimum-

Scheduling scheme	$f_1$	$n_1$	$f_2$	$k_1$	$k_2$	$\rho$	$\mu$
Multilevel-assignment HRR	10	2	20	1	1	0.11	9.5
Single-level-assignment HRR (case 1)	10	2	20	0	11	0.11	44.9
Single-level-assignment HRR (case 2)	10	2	20	2	0	0.2	4
WRR	100	-	-	11	-	0.11	80.9

Table 3.2: Comparison of the minimum-bandwidth properties of round-robin schedulers

bandwidth property of a queue is then

$$\mu = \rho^{-1} \left( \frac{k_1}{f_1} (f_1 - k_1) + \frac{k_2}{f_2} (f_2 - k_2) + \frac{k_2 n_1}{f_1 f_2} (f_1 - n_1) \right) - \rho^{-1} + 1. \quad (3.48)$$

Suppose a queue has a bandwidth requirement of  $\rho^{\text{req}} = 0.11$ . Table 3.2 gives a comparison of the minimum-bandwidth property of HRR and WRR with different bandwidth allotments. Under multilevel-assignment HRR, the queue could have reserved slots  $\{k_1, k_2\} = \{1, 1\}$ , giving an allocated bandwidth of  $\rho = 0.11$  and a minimum-bandwidth property of  $\mu = 9.5$  by (3.48). Under single-level-assignment HRR, this queue could be allocated  $\{k_1, k_2\} = \{0, 11\}$ , giving a minimum-bandwidth property of  $\mu = 44.9$ , which is more than four times that of the multilevel case. A much better minimum-bandwidth property could be achieved with single-level HRR by allocating  $\{k_1, k_2\} = \{2, 0\}$ , but it uses a bandwidth allocation of  $\rho = 0.2$ , which is underbooking. Finally, consider the case where a WRR scheduler is used. To get the same bandwidth granularity here would require a frame size of  $f_1 = 100$ . So, this queue would have  $k_1 = 11$  reserved slots in each frame and a minimum-bandwidth property of  $\mu = 80.9$ , which is eight times that of multilevel-assignment HRR.

Another scheduler which uses a framing structure in a non-work-conserving manner is Stop-and-Go queueing [23]; a variation of Stop-and-Go is [37]. Like WRR, Stop-and-Go has minimum-bandwidth property which is in the order of a frame. The round-robin idling

schemes and Stop-and-Go, as well as idling VirtualClock, produce departure processes that can be very different from that of the work-conserving schedulers (c.f. §4.3).

### 3.3.4 The Leaky Bucket

A leaky bucket also has a minimum-bandwidth property since it controls the time at which cells depart from a source to the network.

**Theorem 3.4** *A leaky bucket which is initially full with bandwidth  $\rho$  has a minimum-bandwidth property of  $\mu = 1 - \rho^{-1}$ .*

This theorem can be understood intuitively. Note from Figure 2.3 that, during a busy period of the leaky bucket queue, the token buffer is being filled at a rate of  $\rho$  tokens/slot, which is exactly the same rate as the reference queue. In other words, the leaky bucket and the reference queue serve at exactly the same rate in a busy period. However, the leaky bucket has a “head start” on the reference queue because there is initially  $\sigma \geq 1$  tokens in the token buffer. So, each cell departs earlier from the leaky bucket than the reference queue; the earliest is  $d_i = a_i + 1$  which achieves the smallest possible minimum-bandwidth property. Note that this theorem states that the minimum-bandwidth property can be negative. The leaky bucket is mentioned here because it is one of the nodes that a cell visits when traveling down a virtual channel; so, it is possible that all the nodes of a virtual channel, including the leaky bucket UPC device, have a minimum-bandwidth property.

### 3.3.5 Summary

The results of this section, §3.3, have shown that many schedulers provide a guaranteed rate of service in very different ways (minimum-bandwidth property). VirtualClock, PGPS,  $W^2FQ$  and  $W^2FQ+$  all have minimum-bandwidth properties that are  $\mu = 0$ .

Start-time Fair Queueing has a minimum-bandwidth property of  $N - \rho_n^{-1}$  which can be negative. Some schedulers, such as idling VirtualClock, SCFQ and the Airport scheduler have minimum-bandwidth properties in the order of  $N$ . HRR has a minimum-bandwidth property that is related to the frame structure. The results in the remainder of this thesis will be discussed in conjunction with the other design goals, namely implementability (c.f. Chapter 6) and idle bandwidth distribution (c.f. §3.7).

### 3.4 The Tightness of the Minimum-Bandwidth Property

The work of this thesis has focused on deterministic or worst-case bounds. For the most part, the bounds given in this thesis are tight, meaning that they are achievable; for example, a situation was given for SCFQ in §3.3.1 which achieves the minimum-bandwidth property of  $\mu = O(N)$ . Even if the bounds are tight, however, they do not describe the statistical distribution, which depends greatly on the statistics of the source, the topology of the network and the statistics of the interfering traffic. To get an idea of what this is for a given situation, simulation can be used. A simulation study was done to study delay of FIFO scheduling in [66]. From the empirical distribution of the delay using FIFO scheduling, it was shown that the delays are typically much smaller than the worst-case bound of PGPS and Stop-and-Go. This results gives the idea that the worst-case bounds are typically much larger than what is experienced. However, the advantage of the worst-case bounds is that they do not depend on difficult to predict factors such as the topology of the network or the statistics of the other sources.

### 3.5 Schedulers without Minimum-Bandwidth Properties

Of the schedulers that have been proposed in the literature, many, including those presented earlier in this chapter, are designed with bandwidth guarantees in mind. However,

it is very easy to find counter-examples of schedulers that cannot provide a minimum-bandwidth property. The simplest one is the FIFO scheduler which simply serves all the cells in the PSN in the order in which they arrive. This scheduler has no way of segregating the input traffic and so a connection which transmits cells to the PSN at a very high rate can effectively starve other connections of service. The following is a more complex scheduler based on a fairness criterion which also cannot provide bandwidth guarantees.

### 3.5.1 The HOL Scheduler

This scheduler is similar to an idea proposed in [63] [62]. The scheduler is like a VFT scheduler, but the timestamp is defined as

$$E_i^n = \max(a_i^n, d_{i-1}^n) + \frac{1}{\rho_n}, \quad (3.49)$$

where  $d_0^n = 0$ . At any time  $t$ , the cell with the smallest timestamp at time  $t-1$  (after all the arrivals and departures) is chosen for service. It was reasoned in [62] that each timestamp represents a deadline that the scheduler should meet in order to give fair service. The scheduler described here uses the deadlines to determine the order of service. Note that the “max” term in (3.49) represents the time at which  $c_i^n$  reaches the head-of-line (HOL) position; hence, it is called the HOL scheduler. This scheduler is described here to show an example of a scheduler without the minimum-bandwidth property.

**Definition 3.3** *HOL-state:* The *HOL-state* at time  $t \in \mathbb{Z}^+$  of a PSN using the HOL scheduler is defined by the set

$$\{E^1(t) - t, E^2(t) - t, \dots, E^N(t) - t\},$$

where  $E^n(t)$  is the timestamp of the HOL cell of queue  $n$  after the arrivals and departures at time  $t$ .

**Lemma 3.4** *Let  $[a, b]$ , where  $a, b \in \mathbb{Z}^+$ , be a period when all queues in a PSN are nonempty. When using the HOL scheduler, the HOL-state at time  $t \in \{a + 1, \dots, b\}$  is completely determined by the HOL-state of the PSN at time  $a$ .*

*Proof:* Note that if the  $n^{\text{th}}$  element is the smallest element in the HOL-state at time  $t - 1$ , then HOL cell in queue  $n$  will be chosen for service at time  $t$ ; i.e., the departure time of the HOL cell in queue  $n$  is  $t$ .

Note also that in the interval  $(a, b]$ , the timestamps of all HOL cells are defined by

$$E_i^n = d_{i-1}^n + \frac{1}{\rho_n} \quad (3.50)$$

since all queues are nonempty. Note that at time  $t = d_{i-1}^n$ , which is when cell  $c_{i-1}^n$  is served and cell  $c_i^n$  reaches the HOL position, the  $n^{\text{th}}$  element in the HOL-state is

$$E_i^n - d_{i-1}^n = \frac{1}{\rho_n}. \quad (3.51)$$

The right-hand side of (3.51) does not depend on the value of  $d_{i-1}^n$ ; so, the HOL-state at any time  $t$  only depends on 1) the HOL-state at time  $t - 1$ , and 2) the cell served at time  $t$ . However, recall that the cell served at time  $t$  was chosen completely based on the HOL-state at time  $t - 1$ . This means that the HOL-state at time  $t$  is completely determined by the HOL-state at time  $t - 1$ .

The lemma then easily follows by induction. □

This lemma basically states that the HOL-states, and hence the ordering of cell departures in a period when all queues are nonempty, are completely determined by the

HOL-state at the beginning of the period.

**Theorem 3.5** *The HOL scheduler does not have a minimum-bandwidth property of  $\mu < \infty$ .*

**Proof:** This proof will show that for a particular choice of  $\{\rho_1, \rho_2, \dots, \rho_N\}$ , there is a set of arrival processes for which the HOL scheduler cannot guarantee a rate  $\rho_n$  to some queue  $n$ ; hence, the HOL scheduler doesn't have a minimum-bandwidth property.

Consider a PSN with arrival processes such that, for all time  $t \geq 1$ , all the queues are nonempty. Suppose that the PSN reaches a particular HOL-state at time  $t$ , and then it reaches that same HOL-state again at time  $t + \tau$ , where  $t, (t + \tau) \in \mathbb{Z}^+$ . Then, by Lemma 3.4, it can be seen that the departure process over  $[t, t + \tau - 1]$  is identical to the departure process over  $[t + \tau, t + 2\tau - 1]$ ,  $[t + 2\tau, t + 3\tau - 1]$ , and so on. In other words, the departure process is "periodic" if any HOL-state repeats itself. In order for the PSN to have a minimum-bandwidth property in this situation, the scheduler must provide "periodic" service to each queue. In other words, the number of cells served over  $[t, t + \tau - 1]$  to queue  $n$  must be *exactly*  $\tau\rho_n$ , for all  $n$ . The following is a counterexample to this.

Consider a PSN with two queues ( $N = 2$ ) with the following bandwidth assignments:  $\rho_1 = 5/8, \rho_2 = 3/8$ . Suppose that both queues are continuously busy starting at time 1 (i.e.,  $a = 1$  in Lemma 3.4). The HOL-state and the departures at each slot are shown in Table 3.3. As can be seen, the state at time 2 is identical to the state at time 5 and at time 8; so, the departure process will be periodic over intervals of  $\tau = 3$  slots. Over the 3 slot interval  $(2, 5]$  two cells from queue 1 and one cell from queue 2 are served. Therefore, the service rate given to queue 1 is  $2/3$  while queue 2 gets  $1/3$ , which are clearly not the same as the allocated rates  $\rho_1 = 5/8$  and  $\rho_2 = 3/8$ .

□

Time	$E^1(t) - t$	$E^2(t) - t$	Queue Served
1	8/5	8/3	-
2	8/5	5/3	1
3	8/5	2/3	1
4	3/5	8/3	2
5	8/5	5/3	1
6	8/5	2/3	1
7	3/5	8/3	2
8	8/5	5/3	1
9	8/5	2/3	1
10	3/5	8/3	2

Table 3.3: Timestamps and service for the HOL scheduler.

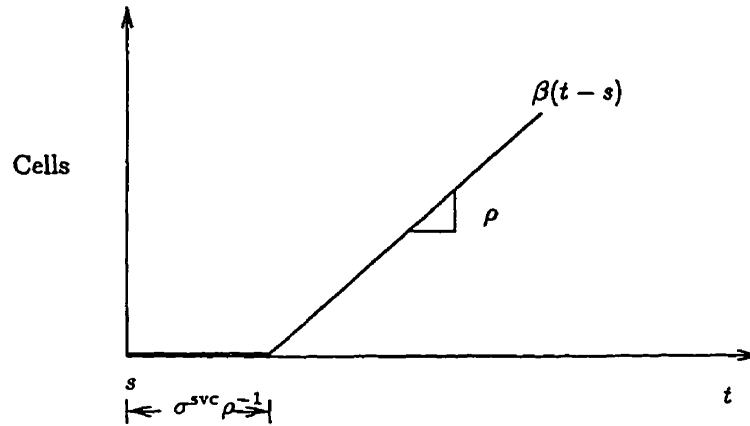
The theorem shows that the HOL scheduler cannot provide bandwidth guarantees efficiently. It was shown in [63], however, that this scheduler can provide “regular” service (or a bandwidth guarantee) if  $\rho_n^{-1}$  is an integer for all  $n \in \{1, 2, \dots, N\}$ , meaning that the HOL scheduler has a minimum-bandwidth property for this special case.

### 3.6 Service Curve

The minimum-bandwidth property is one method of evaluating the usefulness of a scheduler. Here it is compared with another measure proposed in the literature.

**Definition 3.4 Service Curve:** A scheduler guarantees a service curve  $\beta(\cdot)$  if, for any  $t \in \mathbb{Z}^+$ , there exists  $s \leq t$ , where  $s \in \mathbb{Z}^+$ , such that  $Q(s) = 0$  and  $D(s, t) \geq \beta(t - s)$ . A scheduler is said to satisfy a  $(\sigma^{\text{svc}}, \rho)$  service curve if  $\beta(x) = (-\sigma^{\text{svc}} + \rho x)^+$ , where  $\sigma^{\text{svc}}, \rho \geq 0$ .

This definition is taken from [12]; a similar definition is used in [46] to characterize variable rate links (as opposed to being a fixed rate of  $c$  cells/sec). The service curve

Figure 3.4: The  $(\sigma^{svc}, \rho)$  or linear service curve

indicates the amount of service in a busy period of the PSN queue. For simplicity, the  $(\sigma^{svc}, \rho)$  service curve will be called the linear service curve in this thesis. Note that the linear service curve is quite similar to the minimum-bandwidth property: it is based on the concept of a scheduler guaranteeing a rate of service, and there is a notion of a delay of the initial cell in the busy period with  $\sigma^{svc} \rho^{-1}$ . See Figure 3.5. The Airport scheduler [11] was designed to impose a linear service curve. The following gives a comparison of the minimum-bandwidth property and the linear service curve. Before beginning, a point should be made regarding notation here: for the remainder of this thesis, the superscript and subscript notation denoting the index of the queue will be implied.

**Theorem 3.6** *The arrival process, which is  $(\sigma, \rho)$ -constrained, of a PSN queue with a bandwidth allocation of  $\rho$  and minimum-bandwidth property  $\mu \geq 0$  is guaranteed a  $((\sigma \rho^{-1} + \mu + 1)(\rho^{-1} + \mu)^{-1}, (\rho^{-1} + \mu)^{-1})$  service curve.*

**Proof:** Consider an arbitrary scheduler with a minimum-bandwidth property of  $\mu$ . The service curve is only concerned with the busy periods of a queue; so, consider a busy period which begins with the arrival of an arbitrary cell  $i$  at time  $a_i$ ; so,  $s = a_i - 1$  in

Definition 3.4. Assume the busy period is arbitrarily long. The  $(\sigma, \rho)$ -constraint and (3.2) are related in that they both are based on fluid model and that they both use a rate  $\rho$ . Note that  $\sigma \geq 1$ ; so

$$a_i + \sigma\rho^{-1} \geq \mathcal{F}_i \quad (3.52)$$

$$\Rightarrow \mu + \sigma\rho^{-1} \geq \mathcal{F}_i + \mu - a_i \geq d_i - a_i. \quad (3.53)$$

In order for the second cell to arrive at the same busy period as the first, it must be true that

$$a_{i+1} \leq d_i \leq a_i + \sigma\rho^{-1} + \mu. \quad (3.54)$$

Using this bound and (3.52), it can be seen that second cell must depart at

$$d_{i+1} \leq \mathcal{F}_{i+1} + \mu \quad (3.55)$$

$$\leq \max\{a_{i+1}, \mathcal{F}_i\} + \rho^{-1} + \mu \quad (3.56)$$

$$\leq (a_i + \sigma\rho^{-1} + \mu) + \rho^{-1} + \mu \quad (3.57)$$

This argument can be used inductively to show that

$$d_{i+k} \leq a_i + \sigma\rho^{-1} + \mu + k(\rho^{-1} + \mu) \quad (3.58)$$

where the  $(i+k)^{\text{th}}$  cell is a cell in the busy period.

This set of values for the latest possible departures in the busy period gives the smallest possible departure curve  $D(s, t)$ , where  $t$  is an instant in the busy period and  $s = a_i - 1$ . By (3.58), it can be seen that the number of total departures in the interval

$(a_i, t]$  is

$$D(a_i, t) \geq \left( -\frac{\sigma\rho^{-1} + \mu}{\rho^{-1} + \mu} + \frac{t - a_i}{\rho^{-1} + \mu} \right)^+ \quad (3.59)$$

$$\Rightarrow D(s, t) \geq \left( -\frac{\sigma\rho^{-1} + \mu + 1}{\rho^{-1} + \mu} + \frac{t - s}{\rho^{-1} + \mu} \right)^+ = \beta(t - s) \quad (3.60)$$

which, by Definition 3.4, gives the required service curve. Figure 3.5 illustrates this; the curve  $D(s, t)$  that is shown is the smallest possible (worst-case) curve based on (3.58).

□

This clearly shows two significant differences between the information given in the minimum-bandwidth property and the linear service curve. The first is that although the scheduler provides service at a rate of  $\rho$  with a minimum-bandwidth property  $\mu$ , the linear service curve states that the service is at rate of  $(\rho^{-1} + \mu)^{-1}$ . This is shown in Figure 3.5. The implications of this will be further discussed in §4.1. The second difference is that if the source does not have a  $(\sigma, \rho)$  constraint or, in other words,  $\sigma$  is unbounded, then  $\sigma^{\text{svc}}$  is unbounded! So clearly,  $\sigma^{\text{svc}}\rho^{-1}$  and  $\mu$  are not at all similar.

Another measure similar to this is the “(Burstiness) Worst-case Fair Index” [4] [5]. A scheduler satisfies a Worst-case Fair Index of  $\omega$  if

$$D(s, t) \geq \rho(t - s) - \omega \quad (3.61)$$

where  $(s, t]$  is a busy period of the queue. This is almost identical to the definition of the service curve; the only difference is that  $s$  is any time in a busy period while it is the start of the busy period in the definition of the service curve.

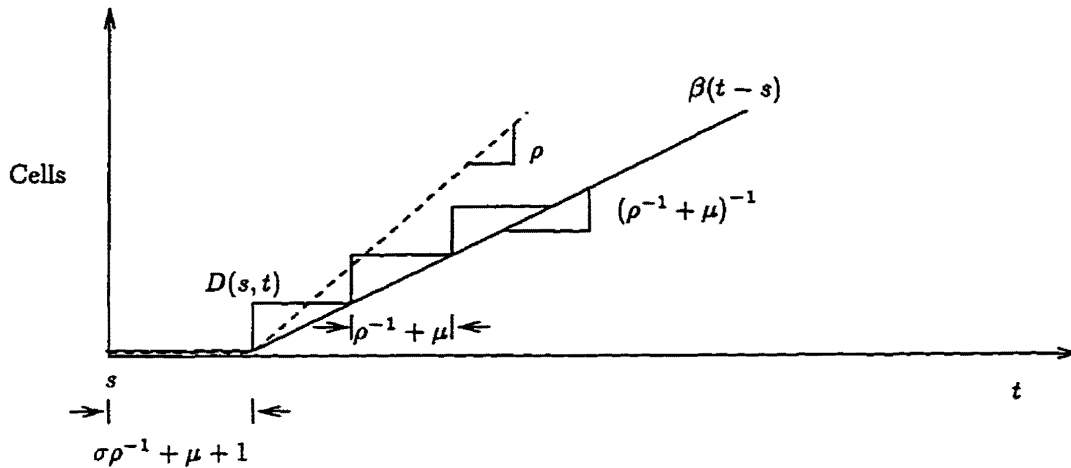


Figure 3.5: The linear service curve of a minimum-bandwidth scheduler

### 3.7 Idle Bandwidth Distribution

The distribution of the idle bandwidth is a design goal that has recently been the focus of much research. Recall that idle bandwidth is created when service slots are unused by the queues. Some schedulers can do this in a direct or *active* manner by leaving specific unused slots which can then be distributed as idle bandwidth by some secondary algorithm; examples of such schedulers are idling round-robin and idling VirtualClock. The primary task of these schedulers is to give to each queue precisely its required bandwidth, and no more. The secondary task of the idling schedulers is to distribute these unused slots to increase the QoS (i.e., decrease cell loss) of connections by serving VBR queues which are experiencing congestion, or to maximize throughput for best-effort traffic by serving it. This type of active distribution is desirable because it allows the network to have better control over the idle bandwidth, which results in more revenue for the network. With active distribution, however, there is the issue of increased complexity, but if complexity is a major issue, simple algorithms to distribute idle bandwidth can be used with the

tradeoff that there is less efficient distribution.

Other schedulers can distribute the idle bandwidth *passively*; for example, PGPS and VirtualClock. Passive idle bandwidth distribution occurs because the scheduling algorithms are work-conserving, and the work-conserving nature is basically a built-in mechanism to distribute the idle bandwidth (so long as there is a nonempty queue). This has the advantage of reducing the need for a secondary scheduler. However, it means that the idle bandwidth distribution is an attribute of scheduler, and the manner in which the idle bandwidth is distributed may not be desirable from the network's point of view. In particular, idle bandwidth will be given to queues which may not need it; e.g., queues supporting CBR connections.

With work-conserving schedulers, better control of the idle bandwidth can be achieved by allocating all of the excess bandwidth  $U = 1 - \sum_{n=1}^N \rho_n$  temporarily to queues that need it; for example, the queues for best-effort traffic. If a queue does not have a bandwidth allocation, it cannot get any idle bandwidth; so, all queues must expect some bandwidth allocation, including those supporting UBR. The excess bandwidth  $U$  may be very small, or even zero, however. In this case, to give reasonable throughput to UBR traffic, some non-negligible bandwidth may have to be permanently allocated to it. This method is inefficient because best-effort traffic is being allocated bandwidth which would otherwise be given to connections with bandwidth guarantee requirements.

If the advantages of using work-conserving schedulers to distribute idle bandwidth are worthwhile, then a *hierarchical* scheduling scheme [4] is recommended. See Figure 3.6. Hierarchical scheduling is similar to HRR. The hierarchical scheduling used here has two levels of scheduling, although more levels are possible. All the queues in the PSN are separated into groups of one or more queues, each of which is served by a secondary scheduler. Each group of queues in the secondary level appears to the primary scheduler as a logical queue; i.e., if the secondary scheduler has a cell to serve, it is regarded as a

nonempty logical queue, and that cell is served by the primary scheduler when it is that logical queue's turn to receive service. Note that a logical queue may be only a single queue, in which case there is no secondary scheduling required. Hierarchical scheduling can be used for hierarchical link sharing, which is where each queue represents connections using an Internet service class such as "telnet" [17].

The queues in the secondary level can be grouped based on their QoS requirements. Figure 3.6 gives a specific example. Every single queue serving CBR traffic is a logical queue. VBR queues are put into a single group. ABR and UBR have their own group. With the hierarchical scheduling scheme proposed here, the primary scheduler is an idling scheduler such as idling HRR or idling VirtualClock. All the idle slots (bandwidth) are distributed to those groups which have best-effort traffic or which can use the idle bandwidth, i.e., groups for VBR traffic. The two-level scheduling scheme is proposed here because it eliminates the problems with using work-conserving schedulers: 1) CBR connections are not given more than their allocated bandwidth because an idling primary scheduler is used, and 2) the idle bandwidth can be directed to best-effort queues by the idling primary scheduler. In addition, two-level scheduling allows the idle bandwidth distribution of work-conserving secondary schedulers to give VBR traffic improved QoS. Hierarchical scheduling has been proposed in [4] using timestamping work-conserving schedulers at both levels; however, such a scheme has less control of the idle bandwidth distribution.

With all hierarchical scheduling schemes, it is necessary to determine if they have a minimum-bandwidth property. It has been shown in [4] that the Worst-case Fair Index of hierarchical schedulers (as described in §3.6) can be expressed as the sum of the Worst-case Fair Indices of the schedulers at each level. This same method can be used to show that minimum-bandwidth property can be expressed as the sum of the minimum-bandwidth properties of each level. Also, if both the primary and secondary schedulers

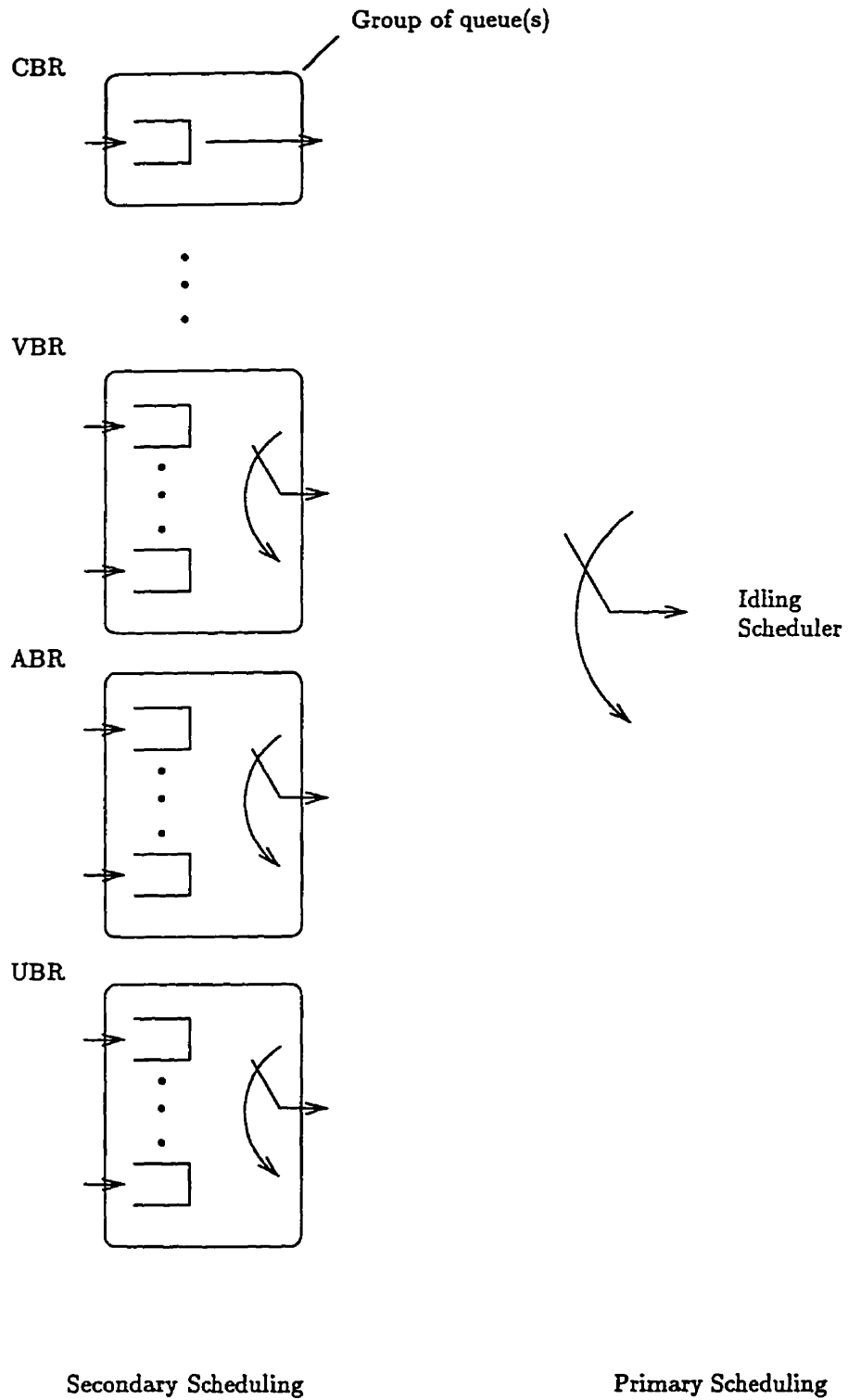


Figure 3.6: Hierarchical scheduling

are idling HRR, then the minimum-bandwidth property of HRR can be used instead.

The way in which work-conserving schedulers can distribute the idle bandwidth in the secondary level is an important issue. The use of schedulers that are “fair” has been advocated. The idea of GPS fairness has been defined earlier; a similar but weaker condition than GPS fairness is the Worst-case Fair Index. This measure is generally an indication of fairness, but in some contexts it can mean bandwidth guarantees. The method used to determine the GPS fairness of a scheduler is usually via a deterministic bound (like the minimum-bandwidth property), but it is unclear how these bounds would translate into a quantity that the network can use to, for example, improve cell loss ratio. One paper [14] takes a probabilistic view of the idle bandwidth by comparing the bandwidth requirement (called the *effective bandwidth*) of an isolated reference queue to a queue in a PSN using a GPS scheduler, both of which have the same arrival process. In this manner, the amount of statistical multiplexing among the queues sharing the GPS scheduler can be quantified. This can then be translated into a lower bandwidth requirement for connections, which would then result in more connections being admitted into the PSN.



## Chapter 4

# Extending the Minimum-Bandwidth Property

In Chapter 3, the notion of a bandwidth guarantee via the minimum-bandwidth property at a single PSN was defined and discussed extensively. This section now brings this into the context of a network. In particular, the usefulness of rate guarantees is shown by demonstrating the existence of an end-to-end delay bound; other results related to the minimum-bandwidth property are given which are very useful for resource management within a virtual channel of PSNs with minimum-bandwidth properties, or “minimum-bandwidth PSNs”.

### 4.1 End-to-End Delay Bound

In an actual network, a delay sensitive source transmits on a virtual channel which is composed of a number of switches (PSNs) in tandem; in each switch it is allocated  $\rho$  bandwidth. For example, a Pulse Code Modulation (PCM) voice connection transmits at regular intervals and has a deadline for each sample to arrive at the destination. In

an ATM network, these samples are “stuffed” into ATM cells which must arrive at the destination by a certain time. This is the crucial application of a scheduler’s ability to provide service rate guarantees: the ability to guarantee a bound on the end-to-end delay through a virtual channel (or virtual circuit) of tandem PSNs.

Consider a virtual channel consisting of one or more PSNs, as shown in Figure 4.1. There are  $K$  PSNs in the virtual channel, and the destination is node  $K + 1$ . The connection is allocated a bandwidth of  $\rho$  at each PSN. Let  $a_i(k)$  and  $d_i(k)$  be the arrival and departure times of cell  $i$  from node  $k$ , respectively. Let  $\{\mathcal{F}_i(k)\}$  be the VC-VFTs based on the arrival process  $\{a_i(k)\}$ . Let  $a_i$  be the arrival time of cell  $i$  to the virtual channel and let  $\{\mathcal{F}_i\}$  be the VC-VFTs based on  $\{a_i\}$ . Note that each PSN may have a different minimum-bandwidth property, denoted by  $\mu_k$  at PSN  $k$ . Finally, let  $\pi_k$  be the propagation delay from the node  $k$  to the next node.

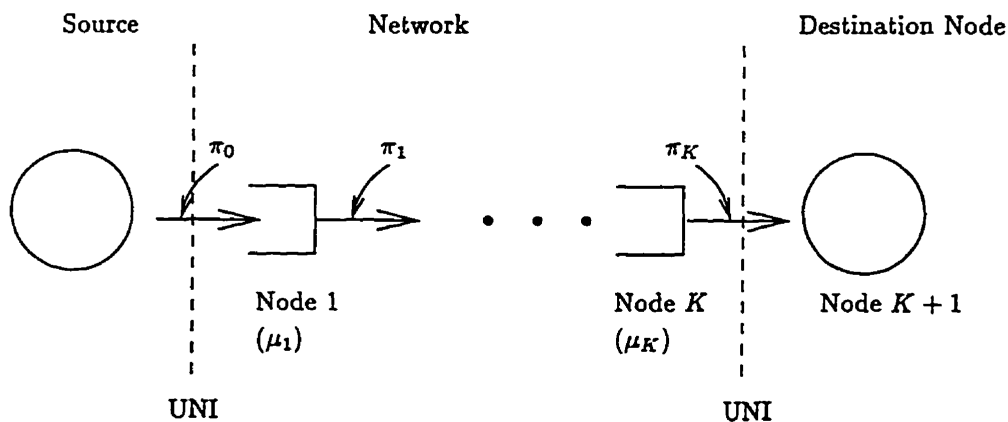


Figure 4.1: A virtual channel of tandem queues

Now suppose that the arrival process of the tandem PSN queues is coupled to an isolated fluid reference queue just as in the single node case; see Figure 4.2.

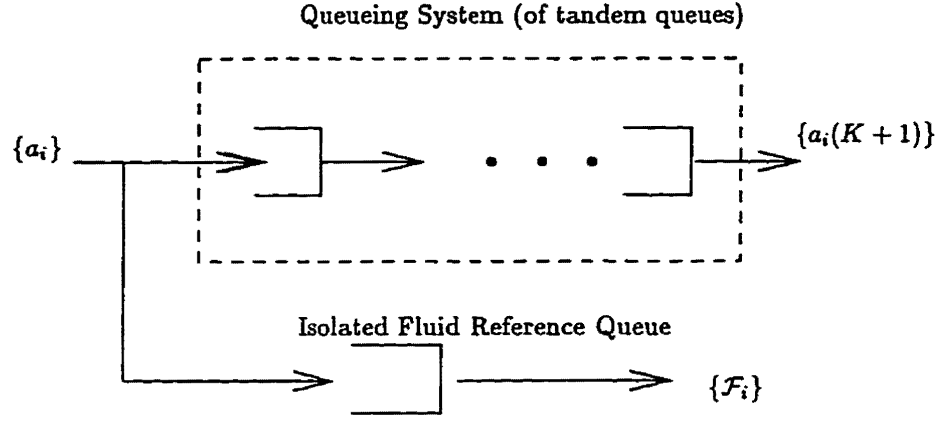


Figure 4.2: Coupling tandem PSN queues to a reference queue

**Theorem 4.1** *The arrival time of cell  $i$  at a node  $k+1$ , where  $k \geq 1$ , is*

$$a_i(k+1) \leq F_i + (k-1)\rho^{-1} + \sum_{l=1}^k \mu_l + \sum_{l=0}^k \pi_l. \quad (4.1)$$

**Proof:** This is a proof by induction. The departure from the first PSN is

$$d_i(1) \leq F_i + \mu_1 + \pi_0 \quad (4.2)$$

by using the minimum-bandwidth property and the propagation delay of  $\pi_0$ . Using this and the fact that the arrival time at the second PSN is  $a_i(2) = d_i(1) + \pi_1$  gives

$$a_i(2) \leq F_i + \mu_1 + \pi_0 + \pi_1; \quad (4.3)$$

so, the theorem is true for  $k = 1$ .

Assuming that the theorem holds for  $k \leq m$ , the following shows that theorem holds

for  $k = m + 1$ . Let  $\widehat{a}_i(m + 1)$  be the value which achieves (4.1) with equality; i.e.,

$$\widehat{a}_i(m + 1) = \mathcal{F}_i + (m - 1)\rho^{-1} + \sum_{l=1}^m \mu_l + \sum_{l=0}^m \pi_l. \quad (4.4)$$

The following temporarily dispenses with the discrete time assumption and assumes that  $\widehat{a}_i(m + 1)$  can be any real value. In this case, all consecutive arrival times to a PSN have a minimum separation of  $\rho^{-1}$ , (i.e.  $\widehat{a}_{i+1}(m + 1) - \widehat{a}_i(m) \geq \rho^{-1}$ ). Combining this with (3.2) gives

$$\mathcal{F}_i(m + 1) = \widehat{a}_i(m + 1) + \rho^{-1}. \quad (4.5)$$

Using the minimum-bandwidth property, the propagation delay of  $\pi_{m+1}$ , (4.5) and (4.4) gives

$$\widehat{a}_i(m + 2) \leq \mathcal{F}_i(m + 1) + \mu_{m+1} + \pi_{m+1} \quad (4.6)$$

$$\leq \widehat{a}_i(m + 1) + \rho^{-1} + \mu_{m+1} + \pi_{m+1} \quad (4.7)$$

$$\leq \mathcal{F}_i + m\rho^{-1} + \sum_{l=1}^{m+1} \mu_l + \sum_{l=0}^{m+1} \pi_l. \quad (4.8)$$

Note that the definition of  $\widehat{a}_i(m + 1)$  implies that  $a_i(m + 1) \leq \widehat{a}_i(m + 1)$ , which in turn implies that the upper bound of  $\widehat{a}_i(m + 2)$  is also the upper bound of  $a_i(m + 2)$ . Therefore,

$$a_i(m + 2) \leq \mathcal{F}_i + m\rho^{-1} + \sum_{l=1}^{m+1} \mu_l + \sum_{l=0}^{m+1} \pi_l. \quad (4.9)$$

□

This theorem was proved independently from the same result in [25]. A similar end-to-end delay bound was derived for VirtualClock in [16]. Note that this theorem relates

the departure times from tandem PSNs to the departure times of the isolated reference queue ( $\{\mathcal{F}_i\}$ ). The reference queue is used in the deterministic bound on the departure times because there is no general information that is known about the source, e.g., a  $(\sigma, \rho)$  constraint, and in such a situation the source may, for example, transmit at a higher rate than  $\rho$ ; so, there clearly can be no constant or absolute bound given. Using this theorem gives a bound on the delay of cell  $i$  (i.e., the time spent by the cell in the network):

$$\delta_i = d_i - a_i \quad (4.10)$$

$$\leq \mathcal{F}_i - a_i + (K - 1)\rho^{-1} + \sum_{l=1}^K \mu_l + \sum_{l=0}^K \pi_l. \quad (4.11)$$

Note that the minimum-bandwidth property adds to the overall network delay, which is intuitive. Using this delay bound, the network can determine whether a virtual channel can satisfy the user's delay requirement. Although the delay bound is somewhat abstract because it uses the reference queue, it can be applied to situations where there is some constraint on the source, such as the  $(\sigma, \rho)$  constraint. Chapter 5 gives an application.

Theorem 4.1 also has an interesting property. Recall that the bound on the departure time from the single node PSN queue, as given by (3.3), also uses a reference queue. In fact, (4.1) and (3.3) are of the same form with the only difference being that the expression for the single node case; i.e., the minimum-bandwidth property, is due only to queueing delays while there is both a queueing and propagation delay component in the multiple node case. Propagation delay is quite different from the queueing delay in that it is constant and is not a function of the scheduler or the arrival process. When the propagation and queueing delay components are separated and the propagation delay component is ignored, the situation is then a single system of PSN queues. See Figure 4.3. This *queueing system* can have a minimum-bandwidth property just as in the single node case. The departure times from PSN queues are denoted by  $\{d_i\}$ .

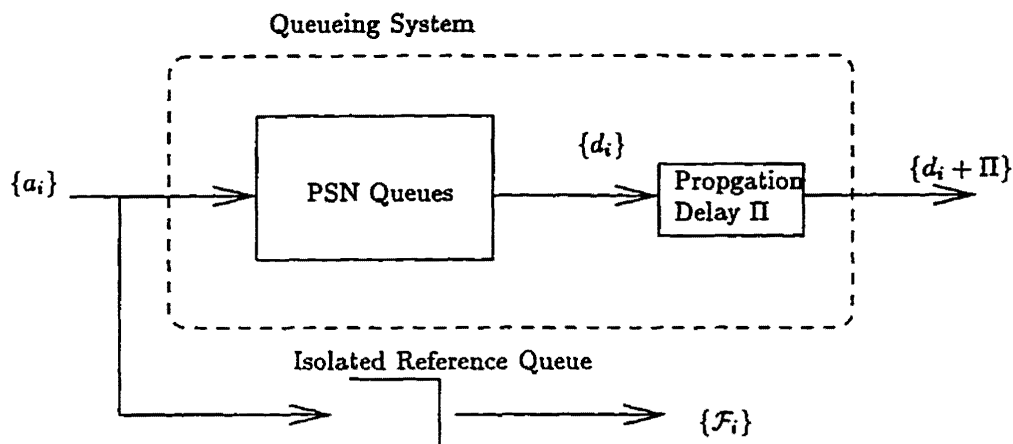


Figure 4.3: Coupling a queueing system to a reference queue

**Definition 4.1** *Effective minimum-bandwidth property:* This is the minimum-bandwidth property of a queueing system neglecting any propagation delays.

The effective minimum-bandwidth property,  $\mu_{\text{eff}}$ , of the queueing system of Figure 4.3 satisfies

$$d_i \leq \mathcal{F}_i + \mu_{\text{eff}}. \quad (4.12)$$

For the virtual channel of Figure 4.1, the total propagation delay is  $\Pi = \sum_{l=0}^K \pi_l$ , and  $d_i = a_i(K+1) - \Pi$ . Combining these values with Theorem 4.1 and (4.12) gives

$$\mu_{\text{eff}} = (K-1)\rho^{-1} + \sum_{l=1}^K \mu_l. \quad (4.13)$$

So, in summary, it has been shown that the departure times from tandem PSN queues can be defined in terms of the minimum-bandwidth property just as in the case of the single PSN queue. In particular, tandem PSN queues can be equivalently represented by a

single PSN queue with minimum-bandwidth property given by (4.12) plus a propagation delay.

Note that the effective minimum-bandwidth property has a  $(K - 1)\rho^{-1}$  term; so, for schedulers with minimum-bandwidth properties that are independent of  $\rho$  (e.g. Virtual-Clock), the delay is larger for connections with small  $\rho$ . This is a disadvantage; e.g., a 64 kbps conversation connection would experience larger delays than a high bandwidth real-time teleconferencing connection. Schedulers which do not have this problem are Start-time Fair Queueing and idling HRR. The reason for this will be given in the following section where the source is assumed to be  $(\sigma, \rho)$ -constrained.

## 4.2 Delay Bound and Buffer Sizing for $(\sigma, \rho)$ -Constrained Sources

The value  $\mathcal{F}_i$  must be used as a reference in the previous section because the cells that enter the virtual channel are assumed to be unconstrained. However, if there is an UPC device which imposes a  $(\sigma, \rho)$  constraint on the arrivals to the network, then an end-to-end delay and buffer size bound exists, as will be shown. The model used in this section is the same as Figure 4.1 except that the source is  $(\sigma, \rho)$ -constrained.

**Theorem 4.2** *If the arrival process of a PSN queue with a minimum-bandwidth property of  $\mu$  and allocated rate  $\rho$  is  $(\sigma, \rho)$ -constrained, a queue size of*

$$B \geq \sigma + \mu\rho + 1 \quad \text{cells} \quad (4.14)$$

*is sufficient to ensure that overflow never occurs.*

Proof: Note that for the fluid reference queue,

$$\Omega(t) \geq \Omega(s) + A(s, t) - (t - s)\rho \quad (4.15)$$

for all  $t > s \geq 0$ , and the equality is achieved when  $(s, t]$  is a busy period. Also, by the definition of the  $(\sigma, \rho)$  constraint (see §2.5),  $\Omega(t) \leq \sigma$ , for all  $t \in \mathbb{R}^+$ . By these two properties,

$$\Omega(a_i) + A(a_i, d_i) - (d_i - a_i)\rho \leq \Omega(d_i) \leq \sigma, \quad (4.16)$$

which implies

$$\Omega(a_i) + A(a_i, d_i) \leq (d_i - \mathcal{F}_i + \mathcal{F}_i - a_i)\rho + \sigma \quad (4.17)$$

$$\leq (\mu + \mathcal{F}_i - a_i)\rho + \sigma, \quad (4.18)$$

where (4.18) is by the minimum-bandwidth property.

Using Lemma 3.2 gives

$$Q(d_i^-) = \Omega(\mathcal{F}_i) + A(\mathcal{F}_i, d_i) + 1 \quad (4.19)$$

$$= \Omega(a_i) + A(a_i, \mathcal{F}_i) - (\mathcal{F}_i - a_i)\rho + A(\mathcal{F}_i, d_i) + 1, \quad (4.20)$$

where (4.20) is by using (4.15) with equality since the interval  $(a_i, \mathcal{F}_i]$  is a busy period of the reference queue. Then, simplifying (4.20) gives

$$Q(d_i^-) = \Omega(a_i) + A(a_i, d_i) - (\mathcal{F}_i - a_i)\rho + 1 \quad (4.21)$$

$$\leq \sigma + \mu\rho + 1, \quad (4.22)$$

where (4.22) is by (4.18). This proves the theorem since the queue occupancy is maximal just prior to departures. Note that this lemma holds even if  $\mathcal{F}_i > d_i$  since in that case  $A(\mathcal{F}_i, d_i) = -A(d_i, \mathcal{F}_i)$ .

□

**Theorem 4.3** *If the arrival process of a PSN queue with a minimum-bandwidth property of  $\mu$  and allocated rate  $\rho$  is  $(\sigma, \rho)$ -constrained, then the departure process is*

$$(\sigma + \mu\rho + 1, \rho) - \text{constrained}$$

Proof: Consider arbitrary  $i, j$  such that  $i > j \geq 1$ . By (3.2)

$$\mathcal{F}_i \geq \mathcal{F}_j + (i - j)\rho^{-1}. \quad (4.23)$$

The  $(\sigma, \rho)$  constraint and (3.2) are related in that they both are based on fluid model and that they both use a rate  $\rho$ . Note that  $\sigma \geq 1$ ; so that

$$a_i + \sigma\rho^{-1} \geq \mathcal{F}_i. \quad (4.24)$$

Combining these two inequalities gives

$$a_i - \mathcal{F}_j \geq (i - j - \sigma)\rho^{-1}. \quad (4.25)$$

Using the fact that  $a_i \leq d_i \leq \mathcal{F}_i + \mu$  and (4.25) gives

$$d_i - d_j \geq a_i - \mathcal{F}_j - \mu \quad (4.26)$$

$$\geq (i - j - \sigma)\rho^{-1} - \mu \quad (4.27)$$

$$\Rightarrow (\sigma + \mu\rho + 1) + (d_i - d_j)\rho \geq i - j + 1. \quad (4.28)$$

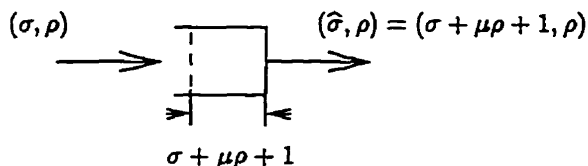


Figure 4.4: Buffer size requirement and burstiness of the departure process of a PSN queue

Note that (4.28) and (2.6) are of the same form; over the length of time between the departure of cell  $i$  and  $j$  from the queue ( $d_i - d_j$ ), there are a total  $(i - j + 1)$  departures from the queue. Hence, the departure process of the queue is  $(\sigma + \mu\rho + 1, \rho)$ -constrained.

□

A similar result for the case of an idealized scheduler which gives a constant maximum delay for each cell is given in [10]. To summarize, Theorem 4.2 and Theorem 4.3 have shown that, given that the arrival process is  $(\sigma, \rho)$ -constrained, there is a finite queue size requirement to ensure overflow never occurs and that the departure process is  $(\hat{\sigma}, \rho)$ -constrained, where  $\hat{\sigma} = \sigma + \mu\rho + 1$ , as shown in Figure 4.4.

At this point, it is possible to derive a bound on the delay of a  $(\sigma, \rho)$ -constrained arrival process that passes through a virtual channel of minimum-bandwidth PSNs.

**Corollary 4.1** *If the arrival process to the network is  $(\sigma, \rho)$ -constrained, the end-to-end delay of cell  $i$  is*

$$\delta_i \leq \sum_{l=1}^K \mu_l + \sum_{l=0}^K \pi_l + (\sigma + K - 1)\rho^{-1}, \quad (4.29)$$

and a queue size of

$$B_k \geq \sigma + k + \rho \sum_{l=1}^k \mu_l \quad \text{cells} \quad (4.30)$$

at PSN  $k$  is sufficient to guarantee that buffer overflow never occurs.

This end-to-end delay portion of this corollary was originally proved in [25]. The corollary is a direct consequence of the previous results in this chapter. The end-to-end delay portion can be proved by combining (4.24) and (4.11). The queue size can be proved in two ways: 1) inductively using Theorem 4.2 and Theorem 4.3, or 2) using the effective minimum-bandwidth property. Similar results were derived for the case of PGPS in [53] and for a number of other schedulers in [67].

As mentioned in the previous section, the end-to-end delay bound is larger for connections with smaller  $\rho$  if, for example, VirtualClock is used. A scheduler which does not have this problem is Start-time Fair Queueing. Assume that the Start-time Fair Queueing is used at every PSN and  $N$  is constant for all PSNs. Given that the source is  $(1, \rho)$ -constrained, the minimum-bandwidth property of  $N - \rho^{-1}$  can then be substituted into Corollary 4.1 to give

$$\delta_i \leq KN + \sum_{l=0}^K \pi_l. \quad (4.31)$$

So, the advantage of a constant delay is evident for a delay-sensitive CBR source: the bound on delay is independent of  $\rho$ .

The previous result can be used to get a simple but significant result.

**Corollary 4.2** *For a  $(\sigma, \rho)$ -constrained source, a constant (i.e., not relative to a reference queue) end-to-end delay bound can be guaranteed if and only if the virtual channel has minimum-bandwidth PSNs.*

**Proof:** A virtual channel of minimum-bandwidth PSNs is a sufficient condition for giving a constant delay bound. This is given by Corollary 4.1.

If any of the PSNs in the virtual channel do not have a minimum-bandwidth property, then clearly the end-to-end delay is not bounded. Therefore, a virtual channel of minimum-bandwidth PSNs is also a necessary condition for giving a constant delay bound.

□

An end-to-end delay bound exists for PSNs that satisfy the linear service curve, which is similar to Corollary 4.1; however, unlike the minimum-bandwidth property, the converse is not true. More precisely, a scheduler which has a minimum-bandwidth property  $\mu$  can satisfy an end-to-end delay bound, but it does not necessarily satisfy a linear service curve.

Corollary 4.2 can be used in the following manner. If a switch designer is to choose a scheduler for all the switches in a network so that the network can guarantee a delay bound, then the designer can simply look for switches which satisfy the minimum-bandwidth property. In addition, if it is known that a certain scheduler can offer delay bounds, then the designer knows that this scheduler must have a minimum-bandwidth property.

### 4.3 Delay Jitter and $(\sigma, \rho)$ Shaping of Departure Processes

As shown in this section, the minimum-bandwidth property can be used to give a bound on the delay. An important issue related to delay is *delay jitter*. Generally, delay jitter is the range of the delay experienced by a cell in the network. Since  $d_i$  satisfies

$$a_i \leq d_i \leq \mathcal{F}_i + \mu_{\text{eff}}, \quad (4.32)$$

the delay jitter is then  $\chi = \mathcal{F}_i - a_i + \mu_{\text{eff}}$ . If the source is  $(\sigma, \rho)$ -constrained, then by (4.24), the delay jitter is

$$\chi = \sigma \rho^{-1} + \mu_{\text{eff}}. \quad (4.33)$$

Note that this clearly indicates the relation between delay jitter and the minimum-bandwidth property. In addition, the expression for the delay of cell  $i$  is  $\delta_i \leq \Pi + \chi$ , by (4.11).

The impact of delay jitter is that it can potentially cause large buffer size requirements at the destination node. Consider a delay-sensitive connection that requires the cells to arrive at the destination with a constant delay bound; e.g., interactive voice or video. For such a connection, playback begins at time  $\chi + \Pi$  since that is the delay bound. In addition, the receive or playback buffer at the destination is where cells are stored before they are consumed by a playback device. Note that they are consumed at the same rate as they are transmitted, but with a delay of  $\chi + \Pi$ . It can be seen that the maximum occupancy of the playback buffer,  $B_P$ , is lower bounded by the maximum number of cells that can be transmitted from the source in an interval of duration  $\chi$ , which is

$$B_P \geq \sigma + \rho\chi \geq 2\sigma + \mu_{\text{eff}} \quad \text{cells} \quad (4.34)$$

by (4.33). It is also true that  $B_P$  is lower bounded by the maximum number of cells that can be transmitted from PSN  $K$  in an interval of duration  $\chi$ , which is

$$B_P \geq \hat{\sigma} + \rho\chi \geq \hat{\sigma} + \sigma + \mu_{\text{eff}} \quad \text{cells}, \quad (4.35)$$

where the departure process of PSN  $K$  is  $(\hat{\sigma}, \rho)$ -constrained; see Figure 4.5. This means that the size of the playback buffer is the lower of the two bounds given by (4.34) and

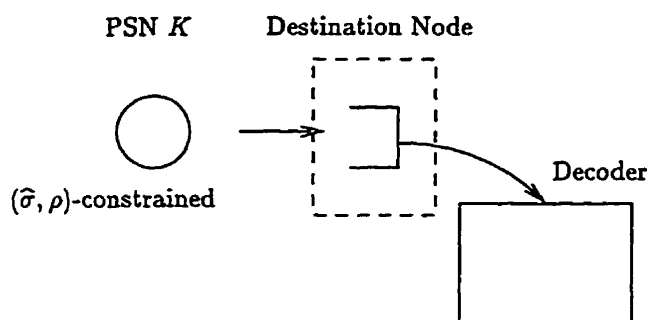


Figure 4.5: Model for the playback buffer

(4.35); i.e., both conditions are true. So,  $B_P$  is a function of both the burstiness of the departure process of PSN  $K$  ( $\hat{\sigma}$ ) and the delay jitter, which in turn is dependent on the burstiness of the source ( $\sigma$ ) and the effective minimum-bandwidth property ( $\mu_{\text{eff}}$ ). For the case of VBR sources,  $\sigma$  can be large, which would contribute to a large playback buffer size requirement; however, by (4.35), it would be less if  $\hat{\sigma} < \sigma$ . The latter part of this section investigates how to control the  $(\sigma, \rho)$  constraints of the departure processes of PSN queues in general.

For connections which are not delay-sensitive such as non-real-time VBR or ABR, there is no longer the need for a large playback buffer since the destination end-user simply consumes cells from the playback buffer whenever it is available; hence, delay jitter is no longer a concern and what can make the playback buffer size large is just  $\hat{\sigma}$ . For CBR connections, where  $\sigma$  is small ( $\sigma$  is just used to account for the lack of synchronization of different switching equipment), a large value for  $\mu_{\text{eff}}$  would make it the dominant term in the playback buffer size, but the value of  $B_P$  here is much smaller than the previously discussed case where  $\sigma$  is large. To reduce the playback buffer size for a CBR connection, Stop-and-Go queueing can be used because the storage of the cells is spread out more evenly over the PSN queues in the network, thereby taking the load

off the playback buffer.

For a process that is  $(\sigma, \rho)$ -constrained and which enters a minimum-bandwidth PSN, Theorem 4.3 gives a general  $(\sigma, \rho)$  constraint for the departure process which follows directly from the minimum-bandwidth property. Recall that the  $(\sigma, \rho)$  constraint is a burstiness envelope which may not necessarily be tight; in other words, the value of  $\sigma$  may be smaller. The value of  $\sigma$  is indeed tight for the work-conserving VFT schedulers. However, with the idling schedulers, such as idling HRR and idling VirtualClock, the bound is a conservative one and a smaller bound is possible. The following shows that the departure process is  $(\hat{\sigma}, \rho)$ -constrained where  $\hat{\sigma}$  can be independent of the input burstiness  $\sigma$ . Recall from Theorem 4.3 that the departure process of the PSN queue is  $(\sigma + \mu\rho + 1, \rho)$ -constrained, which is dependent on the input burstiness,  $\sigma$ .

**Theorem 4.4** *The departure process of a queue served by idling HRR is  $(\xi + 1, \rho)$ -constrained.*

*Proof:* From (3.21), for  $t > s \geq 0$  where  $t, s \in \mathbb{Z}^+$ ,

$$K_l(s, t) \leq (t - s)\varphi_l + \xi_l \quad (4.36)$$

$$\Rightarrow \sum_{l=1}^L K_l(s, t) \leq \sum_{l=1}^L [(t - s)\varphi_l + \xi_l] \quad (4.37)$$

$$\Rightarrow K(s, t) \leq (t - s)\rho + \xi. \quad (4.38)$$

It can be seen that this implies that  $K(s, t) \leq (t - s)\rho + \xi + 1$  for  $t, s \in \mathbb{R}^+$ . The result then follows from the fact that  $D(s, t) \leq K(s, t)$ .

□

The HRR scheduler satisfies this theorem and Theorem 4.3; in other words, the HRR departure process satisfies both envelopes. This in turn means that the envelope can be expressed as the smaller of the two; i.e., the departure is  $(\min(\xi, \sigma + \mu\rho + 1), \rho)$ -constrained

under idling HRR. This is called  $(\sigma, \rho)$  shaping of the departure process. At the very least, given no knowledge of the arrival process, the departure can be determined to be  $(\xi, \rho)$ -constrained. This fact is significant because it allows the buffer size requirements to be small and constant beyond the first PSN; whereas with Corollary 4.1, the buffer sizes increase linearly with the number of hops. Note also that the burstiness, i.e.,  $\xi$ , is totally dependent on the allocated bandwidth  $\rho$  and the HRR frame structure, which is composed of the values  $\{n_1, n_2, \dots, n_L\}$  and  $\{f_1, f_2, \dots, f_L\}$ . Stop-and-Go is similar to round-robin scheduling, and it can control the departure process in a similar way.

Idling VirtualClock can shape the departure process in a similar manner. It was shown in [44, page 65] that the departure process is  $(1 + (N - 2)\rho, \rho)$ -constrained. Here, the burstiness is dependent on the number of queues  $N$  and the allocated bandwidth  $\rho$ .

This type of  $(\sigma, \rho)$  shaping can be considered to be *passive*  $(\sigma, \rho)$  shaping because the value of the burstiness  $\sigma$  is an indirect product of the scheduler design and the bandwidth allocation. As opposed to this, a recent paper proposed an *active*  $(\sigma, \rho)$  shaping device that is a type of *rate-controlled service discipline* [22] which can impose an arbitrary  $(\sigma, \rho)$  constraint. Figure 4.6 shows its model; as can be seen, there is a traffic shaping device, like a leaky bucket, at each queue in the PSN. When using this scheme, however, there is an implementation cost to consider.

#### 4.4 Connections within Virtual Paths

Since the number of expected connections in a network can be quite large, it would be difficult to queue and do VCI translation on each connection individually. To address this problem, ATM has devised the concept of a bundle of connections (VCCs), which is the virtual path connection. In a VPC, connections are multiplexed into one stream at the entry point in a FIFO manner. If two cells arrive at the entry point at the same time, the

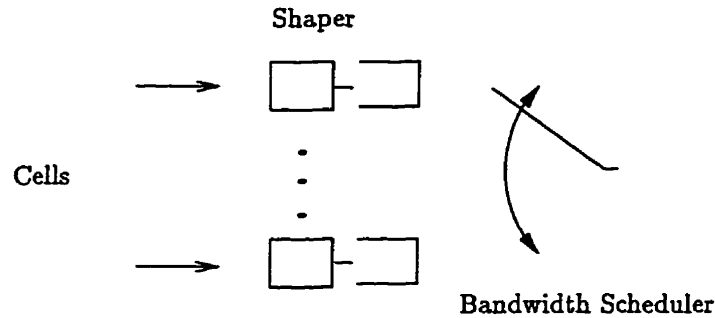


Figure 4.6: Rate-controlled service discipline

ordering can be arbitrary. After the connections are multiplexed together, the entire VPC is treated like a single connection with its own queue in each PSN. This continues until the virtual path terminator is reached, which is where connections are demultiplexed.

Straightforward application of the minimum-bandwidth property can give the same bounds for the whole VPC just as for the single connection considered previously. However, the minimum-bandwidth property is most useful when used to derive bounds with respect to a single connection. So, the goal of this section is to derive the minimum-bandwidth property of a connection which is multiplexed into a VPC.

#### 4.4.1 The Single-Node Case

The first step is to consider the type of multiplexing shown in Figure 4.7. Here, a particular connection is multiplexed into a VPC queue in a PSN. The particular connection in the FIFO queue is analyzed in order to derive the minimum-bandwidth property. It can be seen that to the particular connection, this situation is the same as the two-level hierarchical scheduling described in §3.7 where the secondary scheduler is just a FIFO scheduler.

In order to have a guaranteed rate of service to the particular connection, it is nec-

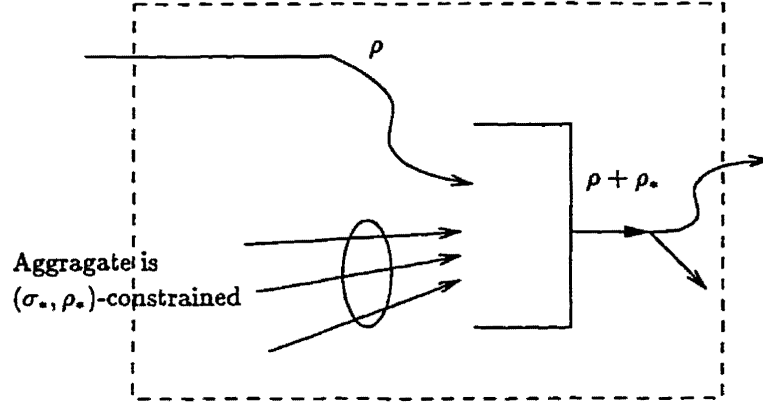


Figure 4.7: Connections sharing a VPC queue

essary to assume that the aggregate process of all the other connections is  $(\sigma_*, \rho_*)$ -constrained. This assumption is a necessary condition for guaranteeing service bandwidth to the particular connection under consideration. It is also natural consequence of leaky bucket traffic shaping; i.e., if all the other connections are policed at the UNI by a leaky bucket and if they all pass through minimum-bandwidth PSNs on the way to the PSN, then they are all  $(\sigma, \rho)$ -constrained. Suppose that the other connections have index  $2, 3, \dots, x$ , and they are  $(\sigma_2, \rho_2)$ ,  $(\sigma_3, \rho_3)$ , ..., and  $(\sigma_x, \rho_x)$ -constrained. It can be shown that the aggregate process produced by all the other connections is  $(\sigma_*, \rho_*)$ -constrained where  $(\sigma_*, \rho_*) = (\sum_{j=2}^x \sigma_j, \sum_{j=2}^x \rho_j)$  [10].

The minimum-bandwidth property is proved in the same way as for the two level hierarchical scheduler in §3.7. Consider cell  $i$  of the connection under consideration. Let  $I(i)$  be the index of this cell within the aggregate arrival process. Let  $\mathcal{F}_{I(i)}^*$  be the VC-VFTs of this cell based on the aggregate arrival process and a service rate of  $\rho_* + \rho$ , and let  $\{\mathcal{F}_i\}$  be the VC-VFTs of the cell based on the connection's own arrival process, with service rate  $\rho$ .

**Lemma 4.1**

$$\mathcal{F}_{I(i)}^a \leq \mathcal{F}_i + \frac{\sigma_*}{\rho + \rho_*} \quad (4.39)$$

**Proof:** Consider a arbitrary cell  $i$  of a particular connection under consideration. Cell  $i$  has an arrival time of  $a_i$ . Let  $s$  be the start of the busy period of the aggregate reference queue when cell  $i$  arrives. Let  $I = \mathcal{F}_i - s$ . Let  $J_a(i)$  and  $J(i)$  be defined as the index of cell  $i$  within the aggregate and the connection's arrival stream, respectively, *within* the busy period. Let  $A_*(s, t)$  be the number of cell arrivals from the other connections in the VPC in the interval  $(s, t]$ .

Note that  $a_i < \mathcal{F}_i$  and that the number of cells from other connections which arrive before cell  $i$  during this busy period is  $A_*(s - 1, a_i)$ . Based on this,

$$J_a(i) = J(i) + A_*(s - 1, a_i). \quad (4.40)$$

Now using the  $(\sigma, \rho)$  constraint gives

$$A_*(s - 1, a_i) \leq I\rho_* + \sigma_* \quad (4.41)$$

In addition, by a simple packing argument (see (3.7)),

$$J(i) \leq I\rho. \quad (4.42)$$

Therefore,

$$J_a(i) = J(i) + A_*(s - 1, a_i) \quad (4.43)$$

$$\leq I\rho + I\rho_* + \sigma_*. \quad (4.44)$$

By the definition of VC-VFTs,

$$\mathcal{F}_{I(i)}^a = s + \frac{J_a(i)}{\rho + \rho_*} \quad (4.45)$$

$$\leq s + \frac{I\rho + I\rho_* + \sigma_*}{\rho + \rho_*} \quad (4.46)$$

$$\leq s + I + \frac{\sigma_*}{\rho + \rho_*} \quad (4.47)$$

$$\leq \mathcal{F}_i + \frac{\sigma_*}{\rho + \rho_*}. \quad (4.48)$$

□

This theorem can be used to give an *effective* minimum-bandwidth property which is given in [44, Theorem 4.3.2] and is restated here for completeness. Let  $\mu_a$  be the minimum-bandwidth property of the VPC queue.

**Theorem 4.5** *The effective minimum-bandwidth property seen by a connection within a VPC queue is*

$$\mu_{\text{eff}} = \frac{\sigma_*}{\rho + \rho_*} + \mu_a. \quad (4.49)$$

**Proof:** Combining the minimum-bandwidth property and Lemma 4.1 gives a bound on the departure time of cell  $i$ :

$$d_i \leq \mathcal{F}_{I(i)}^a + \mu_a \quad (4.50)$$

$$\leq \mathcal{F}_i + \frac{\sigma_*}{\rho + \rho_*} + \mu_a. \quad (4.51)$$

□

#### 4.4.2 The Multiple-Node Case

Consider the case where a VPC traverses several PSNs, as in Figure 4.8. This section shows how to derive the effective minimum-bandwidth property of the VPC at the endpoints shown, where the VPC is from the point of view of the connection. This can be done by simply substituting the effective minimum-bandwidth property of the tandem nodes (see (4.13)) into  $\mu_a$  in Theorem 4.5, where the minimum-bandwidth property is from the point of view of aggregate arrival process. Doing this gives

$$\mu_{\text{eff}} = \frac{\sigma_*}{\rho + \rho_*} + \mu_a \quad (4.52)$$

$$= \frac{\sigma_*}{\rho + \rho_*} + (K - 1)\rho^{-1} + \sum_{l=1}^K \mu_l. \quad (4.53)$$

To summarize, calculating the effective minimum-bandwidth property of a multiple-hop VPC simply involves

1. Deriving the effective minimum-bandwidth property of tandem PSNs which is from the point of view of the VPC, and
2. Deriving the effective minimum-bandwidth property from the point of view of the connection, treating the multiple-hop VPC as a single PSN.

§4.5 gives an example where this is applied.

#### 4.4.3 $(\sigma, \rho)$ Shaping of Departure Processes

Consider again the situation of Figure 4.7. Suppose that the scheduler can shape the departure  $(\sigma, \rho)$  constraint of the queues, just as in §4.3. As mentioned previously, this allows the  $(\sigma, \rho)$  constraint of the departure process to be independent of the burstiness of the arrivals. So, in particular, the aggregate departure process from the VPC queue

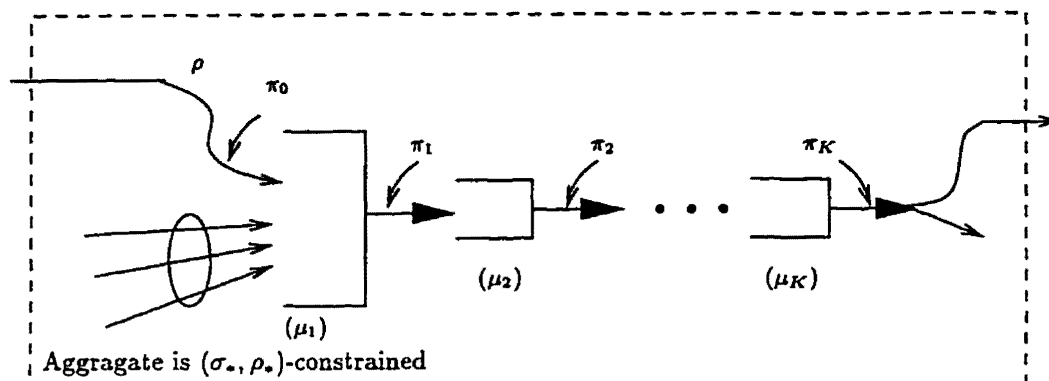


Figure 4.8: A VPC of tandem queues

can be considered to be  $(\sigma + \sigma_*, \rho + \rho_*)$ -constrained, but what is important is the  $(\sigma, \rho)$  constraint of the departure process of the individual connections within the VPC. Recall that the  $(\sigma, \rho)$  constraint determines a bound on the departure process over some interval. In the worst case, it is possible that all the departures of the VPC in some interval are from the particular connection under consideration; so, the departure process of the particular connection is  $(\hat{\sigma}, \hat{\rho})$ -constrained where  $(\hat{\sigma}, \hat{\rho}) = (\sigma + \sigma_*, \rho + \rho_*)$ , which is the same as the aggregate. This result is not useful because only when  $\hat{\rho} = \rho$  can buffer sizing results be derived; in general, this is not possible. However, if the arrival process of the particular connection is  $(\sigma, \rho)$ -constrained, then one can still use the general result of Theorem 4.3 to get the  $(\sigma, \rho)$  constraint of its departure process; i.e.,  $(\hat{\sigma}, \hat{\rho}) = (\sigma + \mu_{e\pi}\rho + 1, \rho)$ .

The lack of an ability to do  $(\sigma, \rho)$  shaping of the departure process can be a major problem if there is feedback. Consider Figure 4.9 where there are three VPCs connecting three VPC queues, and there are three connections using the VPCs. The points at which they enter and leave the VPCs are shown. Consider what is required to calculate the  $(\sigma, \rho)$  constraint of the departure process of connection 1 at queue 1. At queue 1, it shares a VPC with connection 3; so, by Theorem 4.5, the  $(\sigma, \rho)$  constraint of its departure process

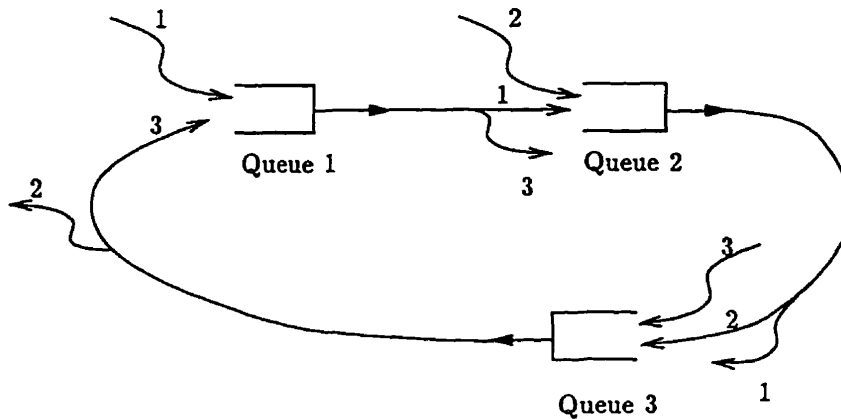


Figure 4.9: A example of feedback when using VPCs

at queue 1 is a function of the  $(\sigma, \rho)$  constraint of the departure process of connection 3 at queue 3. The  $(\sigma, \rho)$  constraint of connection 3 at that point is a function of the  $(\sigma, \rho)$  constraint of the departure process of connection 2 at queue 2, which in turn is a function of the  $(\sigma, \rho)$  constraint of the departure process of connection 1 at queue 1. This is circular definition! This phenomenon was explained in detail in [9]: an example like this was used to show that the effect of feedback with VPCs can cause the departure process to be infinitely bursty. In other words, the queues cannot offer bandwidth guarantees to the connections using the VPCs due to feedback effects. The solution to this problem is simple, however. All that is required is that the connections arrivals be  $(\sigma, \rho)$  shaped at the input to the VPCs; for example, by using a rate-controlled service discipline as in Figure 4.6.

## 4.5 A Virtual Channel Example

This section shows the usefulness of the results of this chapter by taking an example of a virtual channel and deriving its end-to-end delay and buffer sizes. This virtual channel

is shown in Figure 4.10. Note that the connection under consideration goes through an UPC device, a three-hop VPC and then two more PSNs.

First consider the end-to-end delay of the connection. This can be derived by calculating the effective minimum-bandwidth of the entire virtual channel from the source to destination; note this includes the UPC device. Let  $\mu_{i,j}$  be the effective minimum-bandwidth property of the PSNs  $i, i+1, \dots, j$ . By (4.12), it is necessary to calculate the effective minimum-bandwidth property of the virtual channel, which is

$$\mu_{0,5} = 3\rho^{-1} + \mu_0 + \mu_{1,3} + \mu_4 + \mu_5. \quad (4.54)$$

by (4.13). Turning to the three-hop VPC, the aggregate process has an effective minimum-bandwidth property of  $\mu_a = 2(\rho + \rho_*)^{-1} + \mu_1 + \mu_2 + \mu_3$ ; also,  $\sigma_* = \sigma_1 + \sigma_2$  and  $\rho_* = \rho_1 + \rho_2$ . By Theorem 4.5 then,

$$\mu_{1,3} = \frac{\sigma_1 + \sigma_2}{\rho + \rho_1 + \rho_2} + \frac{2}{\rho + \rho_1 + \rho_2} + \mu_1 + \mu_2 + \mu_3. \quad (4.55)$$

Then,

$$\mu_{0,5} = \frac{\sigma_1 + \sigma_2}{\rho + \rho_1 + \rho_2} + \frac{2}{\rho + \rho_1 + \rho_2} + 3\rho^{-1} + \sum_{k=0}^5 \mu_k. \quad (4.56)$$

Finally, the end-to-end delay is

$$\delta_i = d_i - a_i \quad (4.57)$$

$$\leq \mathcal{F}_i - a_i + \mu_{0,5} + \sum_{l=0}^5 \pi_l. \quad (4.58)$$

Suppose that the departure process of the UPC device is  $(\sigma, \rho)$ -constrained; then, the aggregate arrival process of PSN 1 is  $(\sigma + \sigma_1 + \sigma_2, \rho + \rho_1 + \rho_2)$ -constrained. Since the VPC

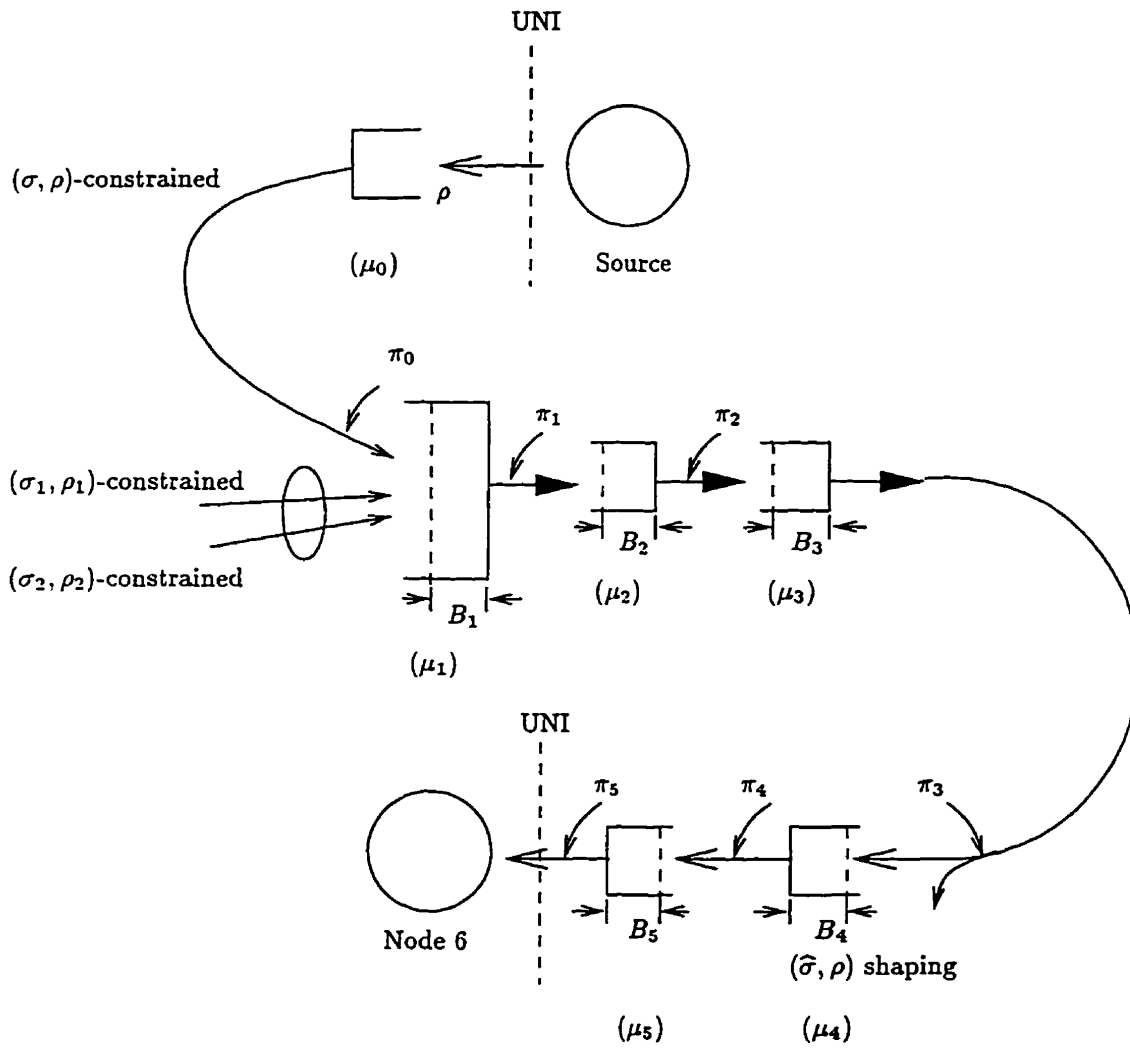


Figure 4.10: An example of a virtual channel

treats this aggregate process in a simple FIFO manner, the results of §4.2 can be applied; i.e., by Corollary 4.1 the buffer size requirement for the VPC at PSN  $k \in \{1, 2, 3\}$  is

$$B_k \geq \sigma + \sigma_1 + \sigma_2 + (\rho + \rho_1 + \rho_2) \sum_{l=1}^k \mu_l + k. \quad (4.59)$$

To get the buffer size at PSN 4, it is necessary to know the  $(\sigma, \rho)$  constraint of the arrival process  $\{a_i(4)\}$ . This can be done by using the effective minimum-bandwidth property of the VPC,  $\mu_{1,3}$ ; so, by Theorem 4.3 the process  $\{a_i(4)\}$  is  $(\sigma + \rho\mu_{1,3} + 1, \rho)$ -constrained. Hence, by Theorem 4.2 the buffer size requirement at PSN 4 is

$$B_4 \geq \sigma + \rho\mu_{1,3} + \mu_4\rho + 2. \quad (4.60)$$

Now note that PSN 4 has  $(\sigma, \rho)$  shaping and the departure process is  $(\hat{\sigma}, \rho)$ -constrained. Hence,

$$B_5 \geq \hat{\sigma} + \mu_5\rho + 1 \quad (4.61)$$

and the process  $\{a_i(6)\}$  (the arrivals to the destination node) is  $(\hat{\sigma} + \mu_5\rho + 1, \rho)$ -constrained.

## Chapter 5

# Application: Transmission of Prerecorded Video

Video service can be classified into two general forms: real-time and prerecorded or stored video. Real-time transmission is a well-known problem and its bursty nature (assuming it is compressed video) is one main reason for using the VBR service. Examples of proposed schemes to handle real-time video traffic are given in [51] [52] [45]. Current video services such as cable television are offering more and more choices to the viewer. There is currently “pay-per-view”, which allows the user a choice of several broadcast channels each showing a different movie. This is a primitive form of “video-on-demand” because the user is only allowed limited choice and the showing times are fixed. These shortcomings are due to the broadcast nature of the service. In the future, it would be desirable to offer the user a large choice of video sequences from a server’s library and have the server transmit whatever video is requested at any time. This type of true video-on-demand has the potential to be a popular and critical service of B-ISDN, especially in view of the success of current retrieval services such as the World Wide Web.

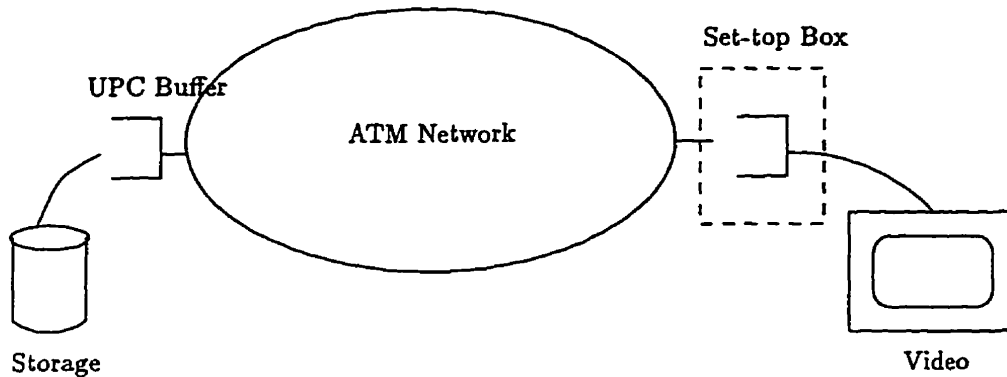


Figure 5.1: Prerecorded video transmission

This chapter describes a resource allocation scheme for PSNs, called piecewise-CBR, which will be able to support this service efficiently. It also discusses a method of provisioning which generates a set of traffic descriptors off-line for the video trace. The methodology in this chapter makes use of the delay and buffer sizing results of Chapter 4.

## 5.1 Properties of the Prerecorded Video Service

The prerecorded video source that is referred to here contains a sequence of compressed video frames; this scheme could be MPEG encoded [20] for example. A framework for prerecorded video transmission over ATM is illustrated in Figure 5.1. A video server fetches prerecorded compressed video sequences or (“traces”) from a disk drive (or some other storage device). The video traces are transmitted by the server through an ATM network. Each video frame is, of course, fragmented into ATM cells before being transmitted, but this delay is assumed to be constant and negligible.

The receiver is a set-top box that, for simplicity here, consists of a playback buffer and a decoder for the received frames. The decoder consumes frames at a constant rate

equal to the rate at which frames are created by the encoder (typically 24 or 30 times per second). In Figure 5.1, the cells are reassembled into frames prior to entering the playback buffer.

The proposed methodology has the following principal properties:

**P1** Lossless and error-free transmission. It is assumed there will be no mechanical and transmission errors in the physical medium.

**P2** Prerecorded and stored awaiting retrieval.

**P3** Long duration; e.g., 2 hrs.

**P4** Playback buffer is small relative to the entire video trace.

**P5** High bandwidth, in the order of Mbps.

**P6** Frames are of highly variable size; i.e., the output from the encoder is bursty.

These properties are the primary motivators for this methodology. Property P1 is a major constraint for the network, but it is necessary since the video is already compressed. Due to this constraint, statistical QoS services such as VBR with some non-negligible cell loss probability is not likely possible. Property P6 is well known and, in fact, compressed video is highly bursty over multiple time-scales [21]. Properties P1, P6 and P5 imply that a per-VCC queueing strategy is a natural choice. Property P2 means that all the data for the video is available before transmission; hence, off-line calculations can be done on the video so that the network can allocate resources in the most efficient manner. Properties P3 and P4 imply that it is not possible to simply download the entire trace to the destination and then initiate playback; transmission is necessary during playback. Property P4 implies that the transmission rate must not be so high that it causes the playback buffer to overflow, but at the same time it must be sufficiently high so that

the playback buffer does not starve. This type of transmission of prerecorded video has characteristics of a file transfer, but it is also similar to real-time transfer in that each frame has a deadline.

## 5.2 Piecewise-CBR

In view of properties given, it is desirable to allocate resources in a CBR-like manner if possible. This section defines a variation of the CBR service class called “piecewise-CBR” and shows that it is an appropriate service scheme for prerecorded video transmission.

Piecewise-CBR gives a long-duration connection CBR service, but allows the bandwidth allocations to vary over intervals of time called *windows*. Let  $W$  be the total number of windows, and let the duration of window  $w$  be  $l_w$  slots. Over window  $w$  the network allocates a CBR service rate of  $\phi_w$  cells/slot,  $w \in \{1, 2, \dots, W\}$ . That is, the connection has the following *schedule* of bandwidth allotments:

$$\vec{\phi} = \{\{\phi_1, l_1\}, \{\phi_2, l_2\}, \dots, \{\phi_W, l_W\}\}.$$

The schedule of bandwidth allotments of a particular connection is known at the time of connection setup. As will be shown below, QoS requirements and minimal network resource allocation goals determine the schedule of bandwidth allotments.

The connection’s maximum bandwidth allotment is

$$\phi_{\max} = \max_{1 \leq w \leq W} \phi_w. \quad (5.1)$$

In piecewise-CBR service, the network reserves  $\phi_{\max}$  for the duration of the connection. Over window  $w$ ,  $\phi_{\max} - \phi_w$  can be temporarily deallocated and made available for best-effort traffic, including ABT. Traffic using this bandwidth will have this bandwidth

removed at the end of the window, and the window should be sufficiently large so that the network's traffic management policy can efficiently exploit this temporarily available bandwidth for ABR and ABT. For example, prior to the beginning of each window, resource management (RM) cells can be used to modify the bandwidth allocations of affected ABR connections in anticipation of changes in excess bandwidth. In this way, ABR connections can efficiently use the excess bandwidth  $\phi_{\max} - \phi_w$ . Also, by reducing the total number of windows,  $W$ , the total overhead involved in executing the piecewise-CBR service is reduced.

A scheme similar to piecewise-CBR is Renegotiated Constant Bit Rate service (RCBR) [28]. RCBR is like piecewise-CBR, but there is no reserved peak bandwidth; so, blocking may occur during an attempt to increase the bandwidth. If blocking occurs, the source would have to lower its rate. Of course, this adds complexity as well as gives uncertain performance.

### 5.3 Resource Provisioning for the Reference Network

The resource provisioning problem for prerecorded video connections will now be addressed. To do this, a virtual channel of PSNs is compared with a fluid reference model, just as in the previous chapters. The reference consists of a single queue without propagation delays. The virtual channel of the video connection consists of an UPC device plus one or more PSNs. In the transmission scheme for prerecorded video, the UPC device acts in essentially the same manner as the reference queue. At a given time, it serves cells at a constant rate of  $\rho$ . See Figure 5.2. Note that the UPC and the reference queue are continuously nonempty because there is always frames to transmit. The destination node has a playback buffer which stores cells awaiting decoding.

It is obvious that the resource provisioning should be done as efficiently as possi-

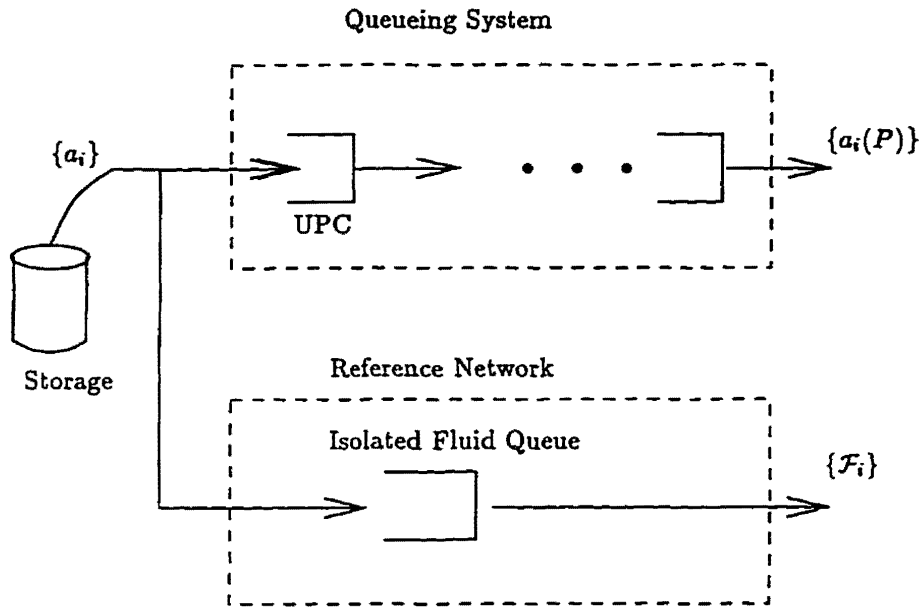


Figure 5.2: The video virtual channel with a reference network

ble; ideally, it should be done in the optimal manner. So, resource provisioning is an optimization problem. Any optimization problem has the following elements:

1. a set of free variables, called traffic descriptors, which can be varied,
2. a set of constraints and
3. a cost function which is to be minimized.

The free variables are parameters which will be given to the network in order to actually transmit the video source; they are also traffic descriptors. They are, namely, playback delay,  $\mathcal{D}_{\mathcal{T}}$ , the number of windows,  $W$ , and the set  $\vec{\phi} = \{\{\phi_1, l_1\}, \{\phi_2, l_2\}, \dots, \{\phi_W, l_W\}\}$ .

The constraints on the optimization problem are the requirements for continuous playback, which are the requirements that the playback buffer should never starve nor

overflow. Let  $\mathcal{S}(t)$  be the total number of cells (work) that are received by the playback buffer of the reference network in the interval  $[0, t]$ . Note that  $\mathcal{S}(\cdot)$  is a continuous function since it is based on a fluid model. Let  $R(t)$  be the cumulative number of cells required by the decoder at time  $t$  when  $\mathcal{D}_{\mathcal{P}} = 0$ ;  $R(\cdot)$  is a piecewise-constant function satisfying

$$R(t) = \sum_{j=1}^i m_j \quad (i-1)\tau \leq t < i\tau \quad (5.2)$$

where  $m_j$  is the size of frame  $j$  and  $\tau$  is the duration of a video frame in units of slots —  $\tau^{-1}$  frames/slot is the rate at which frames are required by the decoder. For  $t \leq 0$ ,  $R(t) = 0$ .

In order to avoid starvation of the playback buffer, the following requirement on the number of transmitted video cells must therefore hold:

$$R(t - \mathcal{D}_{\mathcal{P}}) \leq \mathcal{S}(t), \quad (5.3)$$

where  $\mathcal{D}_{\mathcal{P}} \geq 0$ . In addition, there is a memory size constraint on the playback buffer,  $\mathcal{B}_{\mathcal{P}}$ , as noted in [59]; so, the playback buffer will not overflow if

$$\mathcal{S}(t) \leq R(t - \mathcal{D}_{\mathcal{P}}) + \mathcal{B}_{\mathcal{P}}. \quad (5.4)$$

Figure 5.3 shows graphically what is meant by (5.3) and (5.4). These are the constraints of the optimization problem. Figure 5.3 shows that  $\mathcal{S}(t)$  is a piecewise linear function where  $\phi_i$  is the slope and  $l_i$  is the duration of segment  $i$ .

The cost function which must be minimized consists of two components:

1. the cost in network resources and
2. the initial delay experienced by the end-user,  $\mathcal{D}_{\mathcal{P}}$ .

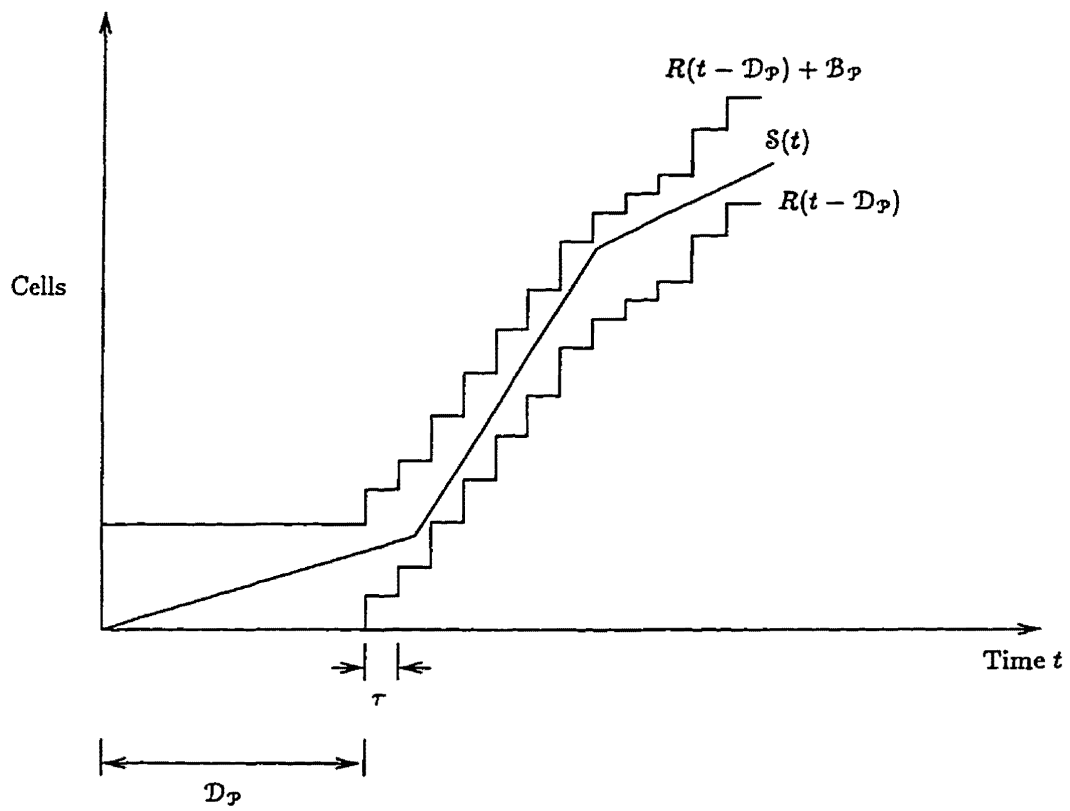


Figure 5.3: Transmission which avoids buffer starvation and overflow.

The following quantifies the cost of the connection from the network's point of view. When a piecewise-CBR virtual channel is setup, the connection reserves  $\phi_{\max}$  bandwidth for the duration of the connection, which is bandwidth that the network can otherwise use to give bandwidth guarantees. Let  $T$  be the duration of the playback so that  $\mathcal{D}_{\mathcal{P}} + T$  is the duration of the connection. The reserved bandwidth  $\phi_{\max}$  is taken from the excess bandwidth of the network. The total *throughput* that this excess bandwidth would offer over the lifetime of the connection is

$$\Lambda_{\text{ex}} = \phi_{\max}(\mathcal{D}_{\mathcal{P}} + T). \quad (5.5)$$

Of course, not all this bandwidth is actually allocated by the connection. The total throughput that is allocated by the connection is

$$\Lambda_{\text{all}} = \bar{\phi}(\mathcal{D}_{\mathcal{P}} + T), \quad (5.6)$$

where  $\bar{\phi}$  is the time average of the  $\phi_i$ 's. The difference  $\Lambda_{\text{ex}} - \Lambda_{\text{all}}$  is the total throughput that can be given to best-effort traffic, including ABT. The values  $\Lambda_{\text{ex}}$  and  $\Lambda_{\text{all}}$  indicate how much work is taken away from the network; hence, they constitute the cost of the connection from the network's point of view. The other component in the overall cost function is the initial playback delay experienced by the end-user (viewer),  $\mathcal{D}_{\mathcal{P}}$ . Note that this is the only user requirement that is flexible — recall that the decoder has no tolerance for loss. The value of this component should reflect the fact that the user can easily tolerate a small delay, but if the delay is excessive then it should become a significant component of the cost function. As an example, the cost function could be defined as

$$c_1\Lambda_{\text{ex}} + c_2\Lambda_{\text{all}} + c_3\mathcal{D}_{\mathcal{P}}, \quad (5.7)$$

where  $c_1$ ,  $c_2$  and  $c_3$  are positive constants.

### 5.3.1 Approaches to Finding $\vec{\phi}$

With the optimization problem defined, the methods of deriving  $\vec{\phi}$  will now be investigated. The first thing to consider is the role and the contribution of traffic descriptors to the cost function. The value of the initial delay,  $\mathcal{D}_{\mathcal{P}}$ , is envisioned to be small; so, it does not severely impact the user's QoS. Furthermore, the prerecorded frames are always available at the source; so, the allocated bandwidth  $\phi_w$  in each window  $w$  is always being fully utilized (except possibly at the very end when there is no longer data to transmit). It follows then that  $\bar{\phi} \approx R(T)/(T + \mathcal{D}_{\mathcal{P}})$ , which is its minimum. Considering this and the fact that  $\mathcal{D}_{\mathcal{P}}$  plays a minor role in the cost function, it is clear the optimization problem is principally a problem of minimizing  $\phi_{\max}$ , which makes sense since, with a small  $\phi_{\max}$ , the transmission is smooth and less bandwidth is reserved.

Given that the main objective in the optimization problem is to reduce  $\phi_{\max}$ , it is now easy to explain the purpose of  $\mathcal{D}_{\mathcal{P}}$ . During playback, some frames can be very large. In this case, transmission of the frames should begin long before they are required by the decoder so that they are transmitted over a long period of time at a low rate. Now if there is a large frame at the beginning of the playback sequence and  $\mathcal{D}_{\mathcal{P}}$  is small, there may be insufficient time before the frame is due to make  $\phi_1$  small. Figure 5.4 shows the effect that the value of  $\mathcal{D}_{\mathcal{P}}$  can have on  $\phi_1$ . Furthermore, if the highest bandwidth allocation is the first window, i.e.,  $\phi_1 = \phi_{\max}$  for a given  $\mathcal{D}_{\mathcal{P}}$ , then increasing the value of  $\mathcal{D}_{\mathcal{P}}$  would decrease  $\phi_{\max}$  which reduces the cost function.

There are a few papers that address similar problems. The work in [59] found the optimal solution for the case where there is no restrictions on the duration of a window. Being optimal in this context implies that the maximum bandwidth allocation  $\phi_{\max}$  is minimized. However, the lack of a lower bound on the window duration is a drawback

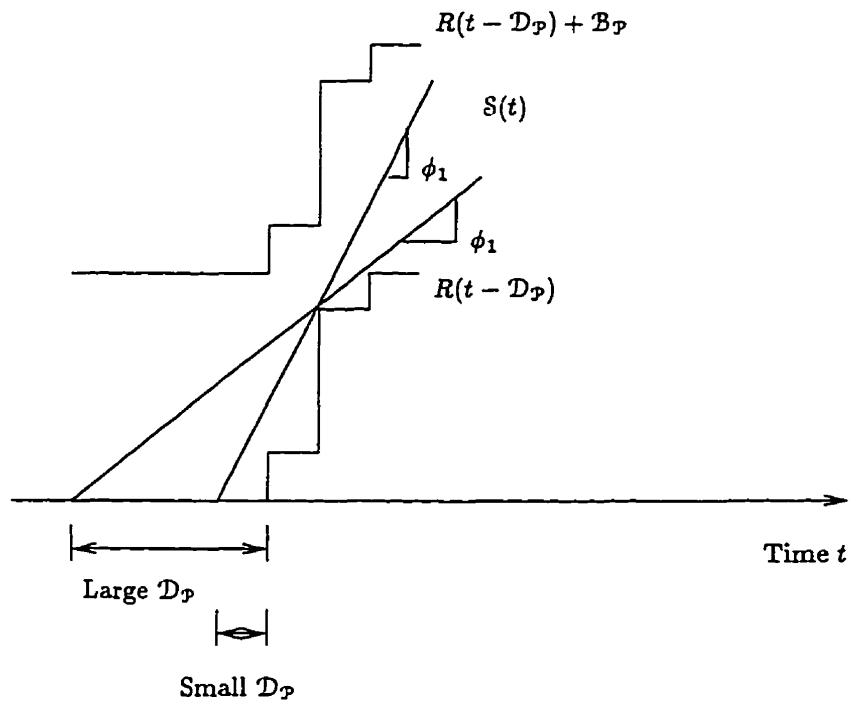


Figure 5.4: Effect of  $D_p$  on  $\phi_1$

since it means that there can be a large number of small windows. In an ATM network, this adds a large overhead in terms of transmission of RM cells, processing at the switches and at the ABR sources. One simple solution to this problem is to use this algorithm but with a larger value for the playback buffer  $\mathcal{B}_P$  so that the minimum window duration is sufficiently large. Better solutions may also be possible by modifying the optimization algorithm of [59]; this is an open problem. An alternative approach to provisioning for prerecorded video is given in [49] where the idea is to optimize given a fixed number of windows  $W$ .

### 5.3.2 The Case for Piecewise-CBR

Piecewise-CBR service has the following attractive features: 1) there is no possibility of call rejection after setup because the maximum bandwidth allotment  $\phi_{\max}$  is reserved and available “on demand” and 2) if the window durations are sufficiently large, the traffic management policies for ABR and ABT traffic may be able to efficiently exploit the excess bandwidth  $\phi_{\max} - \phi_w$  over window  $w$ . An alternative way to transmit prerecorded video in a lossless manner is to use CBR service, which is just the special case of using piecewise-CBR with only one window. The problem with using one or a small number of windows is that the initial playback delay and the playback buffer may have to be quite large. For example, consider the monochrome MPEG-1 “Star Wars” trace of Figure 5.5, which has been used by many researchers.<sup>1</sup> The straight, dotted line in this figure is  $R(T)t/T$ . If playback begins at time zero and a CBR service is used, then  $\mathcal{S}(t) = R(T)t/T$  will give a minimal network bandwidth requirement of  $R(T)/T$ . Note that the maximum playback buffer occupancy in this case would occur at about 4000 sec. To avoid overflow, the required playback buffer size would have to be about 20 Mbytes; this fact was verified

---

<sup>1</sup>This trace is made available via anonymous ftp by M. Garrett of Bellcore and M. Vitterli of U.C. Berkeley at `thumper.bellcore.com` in directory `vbr.video.trace`.

in [49] where it was shown that the minimum possible playback buffer size is 23 Mbytes. In addition, if color is used, the required amount of memory would increase by about an order of magnitude.

In [59], the results of the optimal algorithm were given for the same "Star Wars" trace. The playback delay was set at  $\mathcal{D}_{\mathcal{P}} = 0.5$  sec. (or 12 frames since the frame duration is  $\tau = 24^{-1}$  sec.), and  $\mathcal{B}_{\mathcal{P}}$  was set at 64 Kbytes and 1024 Kbytes for two runs of the algorithm. First, note that if the source was transmitted VBR at the same rate that it was encoded (i.e., a whole frame is transmitted at each period of time  $\tau$ ) then the peak rate of the source is about 4.5 Mbps; meanwhile, the mean rate of the source is  $\bar{\phi} = R(T)/T = 0.37$  Mbps (note that this is roughly the mean rate regardless of whether CBR, VBR or piecewise-CBR is used). It was shown that for  $\mathcal{B}_{\mathcal{P}} = 64$  Kbytes,  $\phi_{\max} = 0.96$  Mbps, and for  $\mathcal{B}_{\mathcal{P}} = 1024$  Kbytes,  $\phi_{\max} = 0.55$  Mbps. Comparing piecewise-CBR for the case of  $\mathcal{B}_{\mathcal{P}} = 1024$  Kbytes with the CBR scenario described previously, one can see that the playback memory requirement is twenty times less for piecewise-CBR, while it reserves a peak rate of  $\phi_{\max} = 0.55$  Mbps which is not significantly wasteful considering that the CBR case uses about 0.37 Mbps, especially in view of the fact that difference of  $\phi_{\max} - \bar{\phi}$  would be given to best-effort traffic. Clearly then, piecewise-CBR with a large number of windows offers significant performance advantages over CBR.

## 5.4 Resource Allocation Over a Virtual Channel

This section describes how to use the traffic descriptor  $(\bar{\phi}, \mathcal{D}_{\mathcal{P}}, \mathcal{B}_{\mathcal{P}})$  to set up a connection across an arbitrary virtual channel of minimum-bandwidth PSNs. To do so, the call admission control entity requires the minimum-bandwidth properties and the propagation delays of each node of the virtual channel. Then, the results of §4.2 can be used to size the queues at each PSN of the virtual channel and to derive the playback initiation time

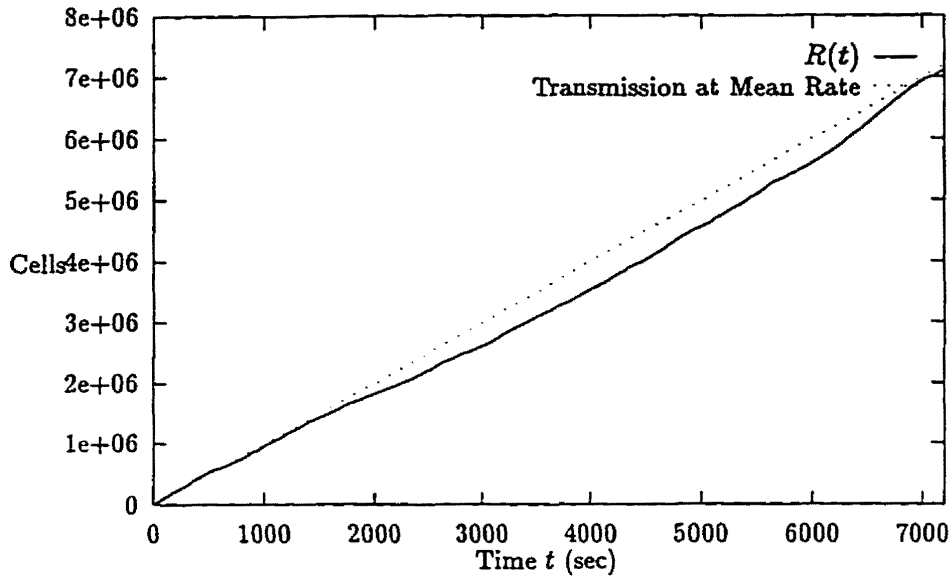


Figure 5.5: MPEG-1 trace of “Star Wars”

$D_P$ . An extension of the results of Chapter 4 will later be given which can determine the playback buffer size  $B_P$ .

The schematic of the arbitrary virtual channel resembles Figure 4.1 with the difference that there is an UPC device which shapes the arrival process of the virtual channel. See Figure 5.6. The UPC device has a minimum-bandwidth property of  $\mu_0$ . In addition, it shapes the arrival of the virtual channel in a manner similar to the  $(\sigma, \rho)$  shaping discussed in §4.3. Let  $S(t)$  be the total number of cells received at the playback buffer in  $[0, t]$ . The UPC (node 0) device performs  $(\sigma, \rho)$  shaping on the source so that  $d_i(0) = \lceil \mathcal{F}_i \rceil$ , which means that the total cell departures from node 0 in the interval  $[0, t]$  is always less than or equal to  $S(t)$ . The implication of this is that

$$S(t) \geq S(t + \Pi), \quad (5.8)$$

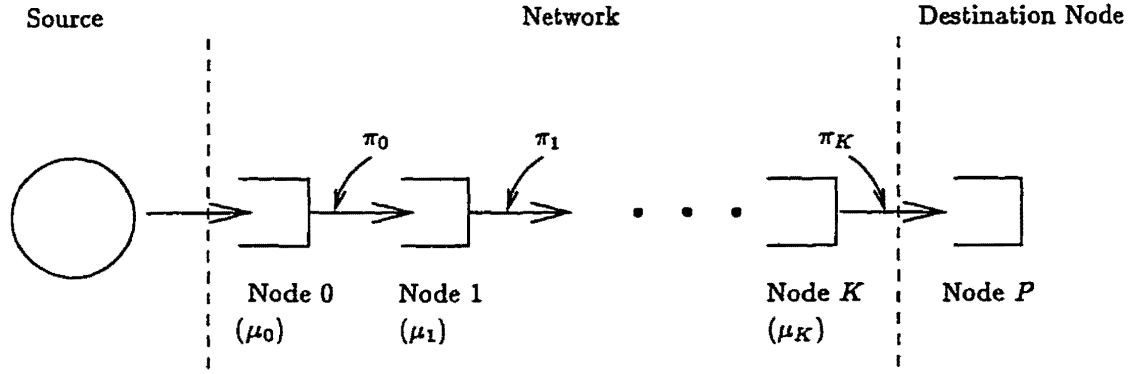


Figure 5.6: The virtual channel for video transmission

where  $\Pi = \sum_{k=0}^K \pi_k$ . Note that  $\mathcal{S}(t)$  is a function easily determined from  $\vec{\phi}$ . This type of shaping is doing roughly the same thing as a leaky bucket with output that is  $(1, \phi_w)$ -constrained during window  $w$ .

So, the number of hops of the virtual channel is  $K$  plus the UPC device. Now let  $P = K + 1$ , so node  $P$  is the playback buffer. It is assumed here that each buffer gives service at a rate of  $\rho = \phi_w$  during window  $w$ . At a given amount of time before the start of the next window; a RM cell will be transmitted to the PSN indicating the impending bandwidth adjustment to  $\rho = \phi_{w+1}$ . The PSN can then react to the impending rate change; for example, by notifying ABR RM cells. The overhead of this RM cell on the transmission bandwidth  $\rho$  is neglected in the following analysis.

Note that the bandwidth allocation  $\rho$  may change over the duration of the connection, which in turn, may alter the minimum-bandwidth property at each node. So here, let  $\mu_k$  be the *maximum* minimum-bandwidth property at node  $k$  over all the bandwidth allocations  $\phi_1, \phi_2, \dots, \phi_W$ , and let  $\phi_{\min} = \min_{1 \leq i \leq W} \phi_i$ . Also, assume that there are at least several cells per window. In the following, variables introduced in Chapter 4 will be used.

**Lemma 5.1** For all cells  $i$ ,

$$\mathcal{F}_i + \Pi \leq a_i(P) \leq \mathcal{F}_i + K\phi_{w(i)}^{-1} + \sum_{k=0}^K \mu_k + \Pi. \quad (5.9)$$

where  $w(i)$  is the window to which cell  $i$  belongs.

This lemma is a direct consequence of Theorem 4.1.

**Theorem 5.1** A queue size of

$$B_k \geq \left[ k\phi_{\min}^{-1} + \sum_{i=0}^k \mu_i \right] \phi_{\max} + 1 \quad \text{cells} \quad (5.10)$$

at PSN  $k \in \{1, 2, \dots, K\}$  is sufficient to ensure that buffer overflow along the virtual channel never occurs.

*Proof:* Consider a PSN  $k$  along the virtual channel. First, let  $\hat{\Pi} = \sum_{l=0}^k \pi_l$  and let  $\widehat{\mu_{\text{eff}}} = k\phi_{\min}^{-1} + \sum_{l=0}^k \mu_l$ ; these values are the propagation delay and the effective minimum-bandwidth property of the queueing system consisting of nodes  $0, 1, 2, \dots, k$  during the window with service rate  $\phi_{\min}$ . Now consider an arbitrary cell  $j$ . Using  $\rho = \phi_{\min}$  in Theorem 4.1, the departure time of cell  $j$  from PSN  $k$  is

$$d_j(k) \leq \mathcal{F}_j + \widehat{\mu_{\text{eff}}} + \hat{\Pi}. \quad (5.11)$$

Let  $l$  be the largest  $i$  such that  $\mathcal{F}_i + \hat{\Pi} \leq d_j(k)$ . It can be seen that the occupancy of the PSN queue at time  $d_j^-(k)$  (which is just before the departure of cell  $j$ ) is at most

$$l - j + 1 \leq (\mathcal{F}_l - \mathcal{F}_j)\phi_{\max} + 1 \quad (5.12)$$

since the departure process of the fluid reference queue, and hence the UPC queue, has

peak rate  $\phi_{\max}$ . Substituting  $\mathcal{F}_l + \hat{\Pi} \leq d_j(k)$  and (5.11) into (5.12) and it can be seen that the occupancy of the PSN queue at time  $d_j^-(k)$  is at most

$$l - j + 1 \leq [(d_j(k) - \hat{\Pi}) - (d_j(k) - \widehat{\mu_{\text{eff}}} - \hat{\Pi})]\phi_{\max} + 1 \quad (5.13)$$

$$= \widehat{\mu_{\text{eff}}}\phi_{\max} + 1. \quad (5.14)$$

Since the occupancy of this queue is maximal immediately prior to departures, the result follows. □

Next, consider the conditions at the playback buffer necessary to provide the video service. Let

$$\mu_{\text{eff}} = K\phi_{\min}^{-1} + \sum_{k=0}^K \mu_k \quad (5.15)$$

be the effective minimum-bandwidth property of the whole virtual channel during the window with service rate  $\phi_{\min}$ .

**Theorem 5.2** *If the playback initiation time is*

$$D_P = \mathcal{D}_{\mathcal{P}} + \mu_{\text{eff}} + \Pi \quad (5.16)$$

*and the playback buffer size is*

$$B_P \geq \mathcal{B}_{\mathcal{P}} + \mu_{\text{eff}}\phi_{\max} \text{ cells,} \quad (5.17)$$

*then the playback buffer will neither starve nor overflow.*

**Proof:** The theorem assumes, of course, that the parameters of the fluid reference model,

$\mathcal{D}_{\mathcal{P}}$  and  $\mathcal{B}_{\mathcal{P}}$ , are known. To first show that the playback buffer will not starve, let  $l(j)$  be the index of the last cell of window  $j$ . The value of  $D_{\mathcal{P}}$  must satisfy

$$(j-1)\tau + D_{\mathcal{P}} \geq a_{l(j)}(P) \quad (5.18)$$

for all  $j = \{1, \dots, W\}$  to ensure that the playback buffer does not starve. Similarly, for the playback buffer of the ideal network to not overflow, the following must hold:

$$(j-1)\tau + \mathcal{D}_{\mathcal{P}} \geq \mathcal{F}_{l(j)} \quad (5.19)$$

$$\Rightarrow (j-1)\tau + \mathcal{D}_{\mathcal{P}} + \mu_{\text{eff}} + \Pi \geq \mathcal{F}_{l(j)} + \mu_{\text{eff}} + \Pi \quad (5.20)$$

$$\geq a_{l(j)}(P) \quad (5.21)$$

Note that (5.21) is due to Lemma 5.1. It can be seen then that the value of  $D_{\mathcal{P}}$  given in (5.16) is sufficient to satisfy (5.18) given (5.21).

The following shows that the playback buffer will not overflow. The occupancy of a (lossless) playback buffer at a given time is the difference between the total cells that have arrived and the total cells that have been decoded; so, the maximum occupancy of the playback buffer in the reference network is

$$\mathcal{B}_{\mathcal{P}} \geq \mathcal{S}(\mathcal{D}_{\mathcal{P}} + i\tau) - R(i\tau) \quad (5.22)$$

$$\geq \mathcal{S}(\mathcal{D}_{\mathcal{P}} + i\tau + \Pi) - R(i\tau) \quad (5.23)$$

by (5.8); this holds for all  $i \in \{0, 1, 2, \dots, W-1\}$ .

Similarly, in order for the playback buffer to not overflow, it is sufficient that

$$B_{\mathcal{P}} \geq S(D_{\mathcal{P}} + i\tau) - R(i\tau). \quad (5.24)$$

The following can find a value for  $B_P$  which satisfies this. By (5.16),

$$S(D_P + i\tau) - R(i\tau) = S(\mathcal{D}_P + \mu_{\text{eff}} + \Pi + i\tau) - R(i\tau) \quad (5.25)$$

$$\leq S(\mathcal{D}_P + i\tau + \Pi) - R(i\tau) + \mu_{\text{eff}}\phi_{\text{max}} \quad (5.26)$$

where (5.26) is due to the fact that the maximum arrival rate of cells to the playback buffer of the ideal reference network is  $\phi_{\text{max}}$  so that  $S(\Delta t + t) \leq S(t) + \phi_{\text{max}}\Delta t$  for all  $\Delta t \geq 0$ . Combining (5.26) and (5.23) gives

$$S(D_P + i\tau) - R(i\tau) \leq \mathcal{B}_P + \mu_{\text{eff}}\phi_{\text{max}}. \quad (5.27)$$

So in conclusion, the value of  $B_P$  in (5.17) is sufficient to satisfy (5.24) given (5.27).

□

To summarize, given the propagation delay and the minimum-bandwidth properties of a virtual channel and given  $(\vec{\phi}, \mathcal{D}_P, \mathcal{B}_P)$  of a video source, the following resources are required along the virtual channel:  $\phi_w$  cells/sec at each hop during window  $w$ , buffer sizes along the virtual channel given by Theorem 5.1, and playback initiation time and playback buffer size given by Theorem 5.2.

One drawback of this approach is that the definition of the initial playback delay  $D_P$  requires that source and destination have synchronized clocks; this would not be required if playback is triggered when a certain number of cells have accumulated in the playback buffer, which is given in [44, Chapter 6]. A paper which complements this [61] describes an approach to resource provisioning for bundles of real-time VBR video teleconferencing connections. Statistical multiplexing at the UPC device is demonstrated for such connections.



## Chapter 6

# Implementation and Complexity

## Issues

The previous chapters have dealt with the theoretical aspects of bandwidth scheduling. A number of results were derived, all related to the Chapter 3 bandwidth scheduling design goals of bandwidth guarantees, good idle bandwidth distribution and efficiency. The one issue that was mentioned in Chapter 3 but not discussed was implementability (at high speeds), which is the purpose of this chapter. In particular, this chapter looks at two issues: implementable bandwidth scheduling algorithms and the use of input-buffered switching, which is more implementable than output-buffered switching.

### 6.1 Implementable Scheduling

One possible architecture of the PSN is shown in Figure 6.1 [8]. It is composed of a main memory unit, which stores cells and associated local information for each cell such as a timestamp and pointer. Queues are divided logically in this memory unit; they can be implemented as a linked list or circular queue. Management of the queues is controlled

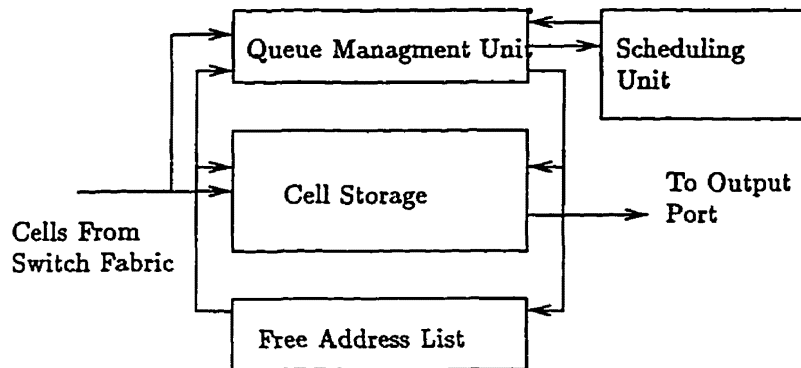


Figure 6.1: An architecture of a PSN

by another unit. One of the tasks of this unit is to calculate the timestamps if they are used. The queue management unit also contains information such as the head and tail pointers of the queues and the current queue occupancy. The free address list is for dynamic memory allocation. The scheduling unit chooses the next cell to serve.

As mentioned previously, there are two main approaches to scheduling. One is to use timestamps and the other is to schedule in a round-robin fashion. With timestamping schedulers that serve in the order of the timestamps, the scheduling unit is a priority queue which contains up to  $N$  head-of-line cells. This can be implemented using a software data structure called a binary heap [30]. Assume for the moment that the scheduler is a single level scheduler (as opposed to a hierarchical scheduler of the type described in §3.7); so, there are  $N$  queues, each with a HOL cell. Enqueue and dequeue operations on the priority queue then have  $O(\log N)$  complexity; in particular, they require about  $k \log_2 N$  operations in the worst case, where  $k$  is a constant. In an output-buffered switch, these  $k \log_2 N$  operations must be executed up to  $F + 1$  times per slot, where  $F$  is the number of input or output ports. As an example let  $F = 16$ , and let  $c$ , the speed of the input and output links, be the equivalent of 2.4 Gbps (this speed is OC-48, a standard rate for ATM).

The duration for each enqueue or dequeue operation (i.e., a slot) is then  $[c(F+1)]^{-1} = 10$  ns. At such high rates, it is very possible that  $N = 32000$ , which is an average of 78 kbps per queue. In this case then, the  $k$  operations must be done in  $(c(F+1) \log_2 N)^{-1} = 0.7$  ns. These  $k$  operations in the binary heap algorithm include at least a read, a write and a compare which must all be done faster than what current memory speeds allow; for example, a very fast memory chip, the Texas Instruments TM5626802, runs at 10 ns [38]. In addition, the values of  $c$  and  $F$  could easily be larger than this example; so, such a design is not implementable at high speeds.

Alternatively, a hardware based algorithm was developed in [8], called a *sequencer*; see Figure 6.2. The sequencer is laid out like a physical queue. The elements of the queue are associated with small blocks in the sequencer, each of which contain the timestamp and identifier of the queue to which it belongs. The rightmost block is the front of the queue, which is the one chosen when it is time to serve the next cell. A comparator exists at each block so that when a new cell enters the sequencer, its timestamp can be simultaneously compared (i.e., in parallel) with every other timestamp in the sequencer. When the position of the new cell is known, a position is opened to the new cell by simultaneously shifting all the necessary cells to the left, and then the timestamp is written. Similarly, when the rightmost timestamp is removed by the scheduler, all remaining timestamps are shifted to the right. All the operations, including the shift, take constant or  $O(1)$  time. The sequencer was implemented in [8], and it was demonstrated to work at 155 Mbps with  $F = 12$  ports. Another implementation of the priority queue which runs in  $O(1)$  time and which does not require custom hardware was given in [56], but the algorithm is only for SCFQ and it is only an approximation of SCFQ.

An additional problem with timestamp scheduling is the complexity of the timestamping operation, which must be done  $F$  times per slot. To calculate the VFT of (3.7) requires two memory accesses for  $\rho_n$  and  $F_{i-1}^n$  as well as an addition and “max” operation;

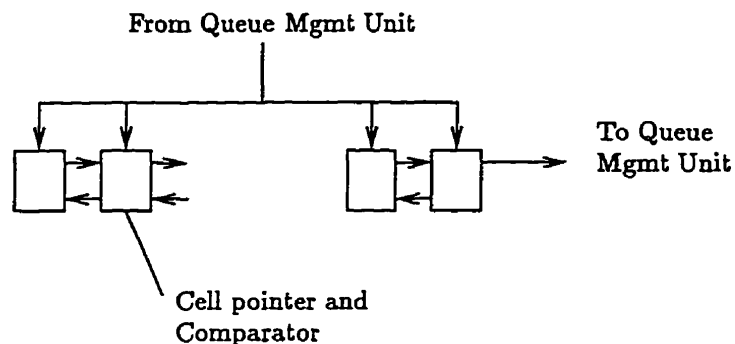


Figure 6.2: A sequencer

all of which must be done  $F$  times. In addition to all this, a virtual time function must be calculated once per slot. With PGPS, at each slot, an *iterative deletion* algorithm that has  $O(N)$  complexity is needed to calculate the virtual time function [43], which is why PGPS is considered to be too complex. Better alternatives are VirtualClock, Start-time Fair Queueing and SCFQ which have virtual time functions that take  $O(1)$  time; the operation is a simple read operation such as the current time for the case of VirtualClock. The  $W^2FQ+$  scheduler requires a priority queue to calculate the virtual time function, which also runs at  $O(\log N)$ . It was argued in [4] that this is implementable since it takes the same amount of time as the binary heap; however, it would be a bottleneck if a sequencer is used.

The memory storage for the timestamps is another cost. It can be seen that the previously mentioned schedulers require a timestamp for each cell; however, an implementation of SCFQ was given in [57] where the timestamping is only done when cells reach the HOL position. This implementation achieves the same ordering as the SCFQ implementation described in §3.3.1; however, it has the advantage that there is only the need to store the timestamps of the HOL cells.

The use of the sequencer gives the fastest possible architecture given that output-

buffering is used. However, using the previous example where the amount of time for each enqueue and dequeue operation is  $[c(F + 1)]^{-1} = 10$  ns, this is still not implementable considering that in this time the virtual time function must be calculated and memory in the cell storage unit must be accessed. This actually brings up a fundamental problem with the overall output-buffering architecture of Figure 1.2: the required memory and processing speeds scale with  $F$ , the number of ports. However, the following section will show that input-buffering is an alternative which requires speeds that do not scale with  $F$ .

Although the implementation problem with timestamp scheduling is considerable, the fact that timestamping schedulers can be work-conserving, meaning they have a built-in method to distribute idle bandwidth, can be very useful for connections that can take advantage of idle bandwidth to improve QoS. Because of the limitations of the software implementation of the priority queue, a sequencer or some similar hardware based algorithm is necessary. However, there is an obvious implementation cost due to the complexity of the hardware design; so, the number of queues that can use timestamps should be limited. The ultimate goal of idle bandwidth distribution is to minimize the bandwidth allocation of connections while still satisfying their QoS, but if the connections are low bandwidth, then the bandwidth savings are not worth the complexity. Therefore, to warrant the use of timestamping schedulers, the connections should be high bandwidth and the number of such connections should be small. Such connections should also be delay and/or loss sensitive and bursty, such as real-time compressed video. The hierarchical scheduling design of Figure 3.6 allows timestamping schedulers to be used in such a manner.

Round-robin scheduling is much simpler to implement and does not require custom hardware. Idling round-robin scheduling should be used for connections that do not necessarily have a special need for fairness. The basic design for using idling round-robin

in conjunction with schedulers that can provide fairness is given in §3.7.

WRR operates by simply moving a pointer at each slot; this takes  $O(1)$  time. HRR uses a separate pointer at each level, as described in [34]: it requires  $O(L)$  pointer operations at each slot, where typically  $L = 2$  or  $3$ , which is of comparable complexity to WRR. Stop-and-Go queueing is similar to round-robin scheduling in that frames are used. Stop-and-Go queueing is best for CBR traffic; however, it is not recommended as an ATM scheduler because it requires that all switches in the network use Stop-and-Go queueing. In general, ATM networks should be large and they should encompass a multitude of applications and equipment from different vendors; so, any ATM design should allow for the use of *inter-operable* equipment. This is not the case with Stop-and-Go queueing.

Multilevel-assignment idling HRR has the balance of good bandwidth granularity and the potential for reasonably good minimum-bandwidth property, as discussed in §3.3.3. However, one point not discussed in the previous description of idling HRR is that it may not always be possible to establish new connections because there may be insufficient slots at one level to accommodate the allocation even though the excess bandwidth is greater than the bandwidth requirement of the new connection. More precisely, recall that there are a total of  $f_l$  slots in the level  $l$  frame. If the number of free slots at each level  $l$  (out of the total  $f_l$  slots) is greater than or equal to  $k_l$  for all  $l$ , then the desired bandwidth allocation is possible. If it is not, the connection establishment attempt may have to be rejected. Note that if this happens, it would not necessarily be due to a lack of available bandwidth, but instead it may be due to an inefficient and inflexible frame structure design.

As an example, consider the frame structure given in the idling HRR example of Table 3.2 where the frame structure is  $\{f_1, f_2\} = \{10, 20\}$  and  $n_1 = 2$ . Suppose that, initially, three connection establishments are attempted, each with a bandwidth allocation of  $\rho^{\text{req}} = 0.17$  cells/slot. So, each connection could use  $\{k_1, k_2\} = \{1, 7\}$ , but the third

connection establishment attempt would fail as the total slots in the level 2 frame is  $f_2 = 20$  and 21 total slots are required. The first solution to this problem is to allocate  $\{k_1, k_2\} = \{2, 0\}$  on the third establishment so that  $\rho = 0.2$  for this connection, which is underbooking by 0.03 cells/slot. This inefficiency can be avoided, however, with a flexible frame structure; i.e., one where the frame size  $f_l$  can increase and decrease to suit the changing needs of connections. This section investigates how this can be done.

The most important condition when changing the frame size is that the bandwidth allocation of existing connections must remain constant. Recall that the bandwidth allocation of a given connection is

$$\rho = \sum_{l=1}^L k_l \frac{n_1 n_2 \dots n_{l-1}}{f_1 f_2 \dots f_{l-1} f_l} \quad (6.1)$$

Suppose that the level  $k$  frame size is to be changed from  $f_k$  to  $\widehat{f}_k$ . Obviously, this would change the bandwidth allocation of the connection, given that everything else is unaltered. However, if the number of slots reserved for level  $k$  frames in the level  $k - 1$  frames can also be changed from  $n_{k-1}$  to  $\widehat{n}_{k-1}$ , then the new bandwidth allocation,

$$\widehat{\rho} = \sum_{l=1}^L k_l \frac{n_1 n_2 \dots \widehat{n}_{k-1} \dots n_{l-1}}{f_1 f_2 \dots \widehat{f}_k \dots f_l} \quad (6.2)$$

would be equal to  $\rho$  if

$$\frac{f_k}{n_{k-1}} = \frac{\widehat{f}_k}{\widehat{n}_{k-1}}. \quad (6.3)$$

This fact was first mentioned in [40].

To illustrate how resizing frames may work, consider the previous example where there are three connections with required allocations  $\{k_1, k_2\} = \{1, 7\}$  and initially,  $\{f_1, f_2\} = \{10, 20\}$  and  $n_1 = 2$ . The scheduler may do the following. First, two connections are

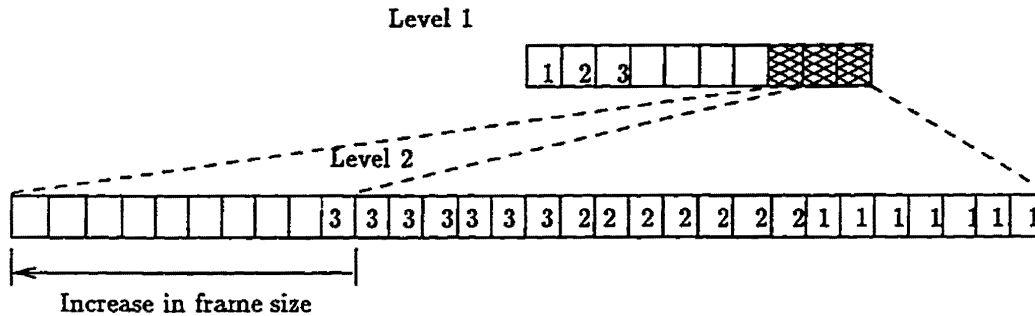


Figure 6.3: HRR frame structure after three connections are established

established. Then, by the condition of (6.3), the level 2 frame is enlarged by setting  $n_2 = 3$  and  $f_2 = 30$ , and finally, the third connection is established. Figure 6.3 shows this; the dashed lines indicate the frame sizes before and after the enlarging.

This example shows that it is possible to retain the bandwidth allocations. However, consider what could possibly happen next: the second connection is terminated leaving the frame structure as shown in Figure 6.4. It is desirable here to decrease the frame size back to  $n_2 = 2$  and  $f_2 = 20$  in order to make the maximum number of slots available to level 1 and, as need be, increase the size of the level 2 frame again when needed. However, this is not possible with idling HRR because the positions of slots used to serve a connection in a frame are fixed; in other words, there is a fragmentation problem. This problem can be avoided if non-idling HRR is used; however, there would be less control over the idle bandwidth, and there is no  $(\sigma, \rho)$  shaping of departure processes which means increased burstiness at the output. To get around this problem, a hybrid version of the idling and non-idling methodology is proposed here.

Recall that idling and non-idling round-robin schedulers differ in that the non-idling version may cycle through all the level  $l$  slots in less than  $f_l$  slots since unused slots are skipped over; so, the size of a frame at any given time is typically less than the allocated

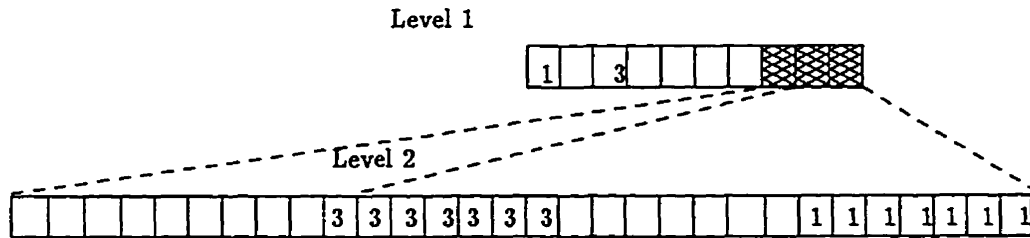


Figure 6.4: HRR frame structure after the second connection is terminated

frame size of  $f_l$  slots. With the idling scheduler, the time it takes to cycle through the queues is exactly  $f_l$  slots since the unused slots are actively distributed as idle bandwidth. To avoid the fragmentation problem of idling round-robin, a third type of round-robin scheduling called *semi-idling* round-robin is proposed in which the duration of a frame is constant, but the allocated slots can be served anywhere in the frame. Figure 6.5 shows how five cells from three connections can be served by one frame in an idling, semi-idling and non-idling WRR schemes. As the name suggests, semi-idling round-robin is in between the idling and non-idling versions. It is similar to idling round-robin because the frame size over a frame is always fixed; so, over a long interval, the amount of service is at most the allocated rate  $\rho$ . It is also like non-idling round-robin because there is no notion of a fixed slot for a connection, each connection is simply given a fixed number of slots in a frame which can be placed anywhere in any given frame. Doing this means that the semi-idling method is more flexible than idling round-robin but still allows the unused slots to be actively distributed as idle bandwidth.

Referring back to the example, it is possible then to arrange the frames as shown in Figure 6.6 after connection 2 is terminated. Note there is no fragmentation problem on level 2 and one more slot in level 1 is available for resource allocation. Of course, this slot can be returned to level 2 whenever needed. The semi-idling methodology means that,



bandwidth property of idling HRR, which is

$$\mu = \rho^{-1} \left( \sum_{l=1}^L \xi_l - 1 \right) + 1, \quad (6.4)$$

where the value of  $\xi_l$  is by Lemma 3.1. Semi-idling HRR has the same minimum-bandwidth property except that the expression for  $\xi_l$  is different.

**Lemma 6.1** *When using semi-idling HRR, for all  $l \in \{1, 2, \dots, L\}$ ,*

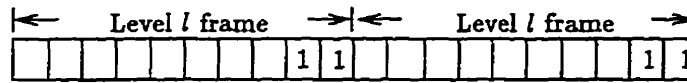
$$\xi_l \leq 2 \left( \frac{k_l}{f_l} (f_l - k_l) + \sum_{i=1}^{l-1} \frac{k_l n_{l-i} \cdots n_{l-1}}{f_{l-i} \cdots f_l} (f_{l-i} - n_{l-i}) \right). \quad (6.5)$$

So,  $\xi_l$  is twice as large for semi-idling HRR as for idling HRR, and this can be proven in the following manner. Recall that Lemma 3.1 was proven using the fact that  $k_l$  is the largest possible “clump” of level  $l$  service slots that can be given to the queue. With semi-idling HRR, the largest possible clump is  $2k_l$  slots, as shown in Figure 6.7 for  $k_l = 2$ . This implies that  $\xi_l$  for semi-idling HRR is double that of idling HRR. This implies that the minimum-bandwidth property of semi-idling HRR is about twice as large as that of idling HRR, but this is a minor penalty for the increased flexibility in allocating bandwidth.

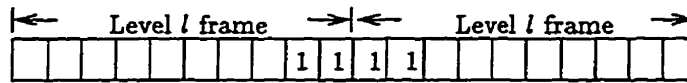
In conclusion, this thesis views multilevel semi-idling HRR as the best general purpose scheduler based on the grounds of flexibly (efficiency), reasonably good minimum-bandwidth property, implementability and good idle bandwidth distribution.

## 6.2 Input-buffering as an Alternative to Output-Buffering

The previous chapters have discussed a framework for bandwidth guarantees using Figure 1.2 as the model for the output-buffered switch and the scheduler. This section considers this model in detail.



(a)



(b)

Figure 6.7: The clumping effect of idling (a) and semi-idling (b) HRR

Each output port PSN contains a physical block of memory for the storage of cells (although there are separate logical queues for VCCs and VPCs within this block of memory). This block of memory is connected to a single bus; hence, reads and writes must be done one at a time. At a given slot, it is possible that all the input port cells must be routed to the same output port, as shown in Figure 6.8a. In order to write all the cells to memory, it is necessary then to *speed-up* the switch from a speed of  $c$  to  $cF$ . This imposes severe demands on the hardware because the net memory bandwidth must be  $(F + 1)c$  cells/sec, which is 10 ns for each read and write using the previous example of  $F = 16$  and  $c$  being the equivalent of 2.4 Gbps; i.e., 10 ns is required to read and write 53 bytes! Recall that a fast memory chip previously described runs at 10 ns; so, this pushes the design to the limit of technology and  $F$  could easily be larger. This example clearly shows that, fundamentally, output-buffering using a single memory block for each output port is *not* implementable.

Now consider the situation of Figure 6.8b. There are  $F$  cells switched in the slot just as in Figure 6.8a, but these cells are all destined for different output ports. So, for this slot, the memory bandwidth required is only twice that of the input and output links (1

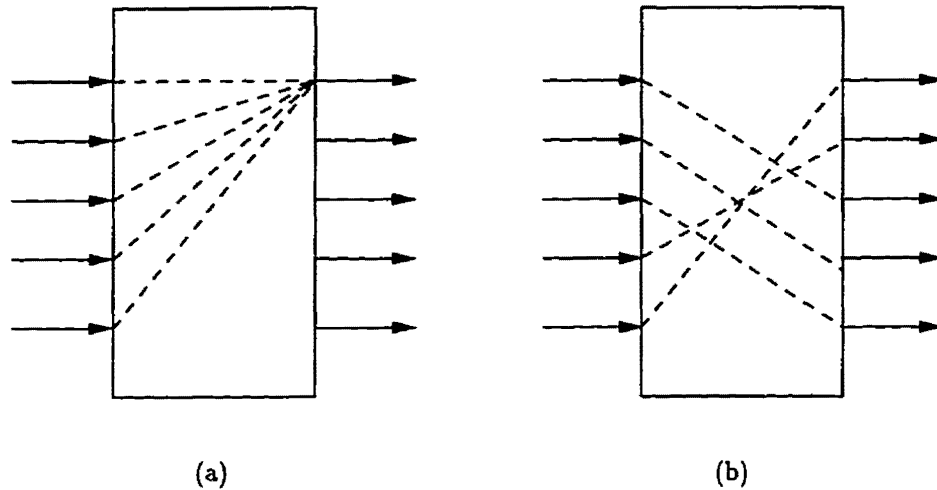


Figure 6.8: The worst (a) and best (b) case for routing

write and 1 read) or  $2c$  cells/sec; so, if  $c$  is equal to 2.4 Gbps, there is 88 ns to read and write each cell.

In order to study implementable designs in this section, it is assumed that the switch fabrics are non-blocking, and they can only route one cell to one output port at each slot. Recall that a non-blocking switch fabric is always capable of routing all cells in a slot provided that they are all destined for different output ports; the crossbar is one example of a non-blocking switch [3] [60].

The situation of Figure 6.8a requires output-buffered switching if cells are not to be dropped, but input-buffering instead of output-buffering can be used if there is at most one read and/or write to an input and output port at each slot. In this section, it will be assumed that an input-buffered switch is a non-blocking switch where the input and output ports can only perform one read and one write at each cell time. The situation where two cells contend for the same output port in input-buffered switches is called contention and *arbitration* is needed to resolve it so that at most one cell is routed to

an output port at any given time. It should be noted here that any ordering of cells for transmission in an input-buffered switch can also be achieved by the output-buffered switch. No matter what arbitration scheme is used, however, the input-buffered switch can never perform better than the output-buffered switch. An early comparison of input versus output-buffered switches was given in [42]. It was shown that, when all cells at an input port are stored into a single FIFO queue, the problem of HOL-blocking can degrade throughput. However, there is no HOL-blocking in the input-buffered switch considered here because each connection (or VPC) is given its own queue. In particular, this means that queues are sorted according to their output port designation; hence, there is no HOL blocking. Nonetheless, the throughput of the input-buffered switch is less than that of an output-buffered switch; an example is shown in Figure 6.9. There is a  $2 \times 2$  input-buffered switch with four cell arrivals in three slots. In the figure, each cell has two numbers indicating the input and destined output port. The output of the switch is shown in the figure. At the first slot, two cells contend for the same output port. The switch has no knowledge of future arrivals, so say it chooses input port 2. At the arrival at the second slot, there are two cells queued in input port 1 that wish to go to different output ports, but due to the 1 read/write limitation, only one of the two cells is chosen. If an output-buffered switch is used, however, all cells are stored in the output port awaiting transmission; so, both cells would be served in this slot. This example shows that there is a throughput difference of three cells with output-buffering versus two cells with input-buffering.

Despite these problems, one may still question whether it is possible to design an arbitration algorithm that can give an input-buffered packet switch almost the same type of performance as an output-buffered switch. There are a number of papers in the literature that discuss performance generally in terms of throughput [47] [48] [3, §3.4] [29, §7.1.1.4]. On the other hand, circuit switching is well-known and can provide bandwidth

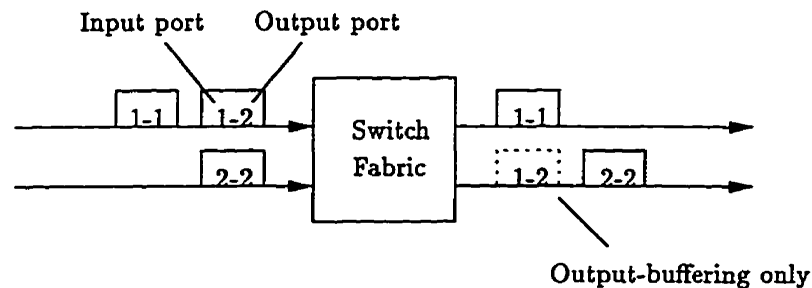


Figure 6.9: Comparison of input and output-buffering

guarantees in a switch where there is at most one read and/or write at an input and output port at each slot. In theory, such circuit-switching designs are based on Clos networks [31] and can be implemented as TST (time-space-time division multiplexing) switches. TST switches have applications in telephone switching networks. Note that the input stream in the telephone switch is TDM and no queueing is necessary beyond the minimal amount that is needed to perform time division multiplexing.

In an integrated packet switching network such as ATM, both bandwidth guarantees and throughput to best-effort traffic must be provided. This can be done by combining the above methods, as first given in [1] and more recently by G. Kesidis in [35]. The basic design here is the same as a TST switch. A time-division multiplexer before each input port feeds the cells to each input port in using a frame structure, just like idling round-robin. The space-division multiplexer then performs non-blocking switching. The last TDM stage is "null"; i.e. it does not alter the order, as shown in Figure 6.10. Note that the PSN has now been moved from the output ports to the input ports.

The key stage of the TST is the initial TDM or round-robin stage. The arbitration is used to schedule the cells that enter the switch fabric. The purpose of arbitration is similar to the PSN scheduling in an output port PSN, but there are key differences. Recall that a PSN scheduler chooses a queue and when a queue is chosen, its HOL cell

is served. The arbitration algorithm chooses output ports, however, and when an output port is chosen, the next cell in line for that output port is served. How to choose the next cell will be discussed later. The difference lies in the fact that the arbitration does not work at the connection level. Instead, all arrivals that originate from the same input port and are destined for the same output port are lumped together as a “bundle”. In an  $F \times F$  switch, there are  $F^2$  of these bundles, one for each input/output port pair. With each input/output pair, there is a required bandwidth guarantee, which is the sum of the bandwidth allocations of all connections that have the same input port and are destined for the same output port. The arbitration algorithm must ensure that

1. No two input ports are routed to the same output port at the same time, due to the memory bandwidth limitation, and
2. The required bandwidth can be guaranteed to each input/output pair.

The allocation of slots in the  $F$  input port arbitration frames can be represented by an  $F \times F$  assignment matrix where  $M_{i,j}$  is the number of slots in a frame that must be routed from input port  $i$  to output port  $j$ . Let  $f$  be the size of the arbitration frames. The following is an example with  $F = 3$  and  $f = 6$ :

$$M = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 2 & 2 \\ 1 & 1 & 4 \end{bmatrix} \quad (6.6)$$

Note that row  $i$  of  $M$  shows the allocated slots of input port  $i$ , and column  $i$  shows the allocated slots of output port  $i$ . Recall that any bandwidth allocation scheme for guaranteeing bandwidth must satisfy the no-overbooking condition. The allocated bandwidth for cells from input port  $i$  to output port  $j$  is  $M_{i,j}/f$  cells/slot. So, the no-overbooking

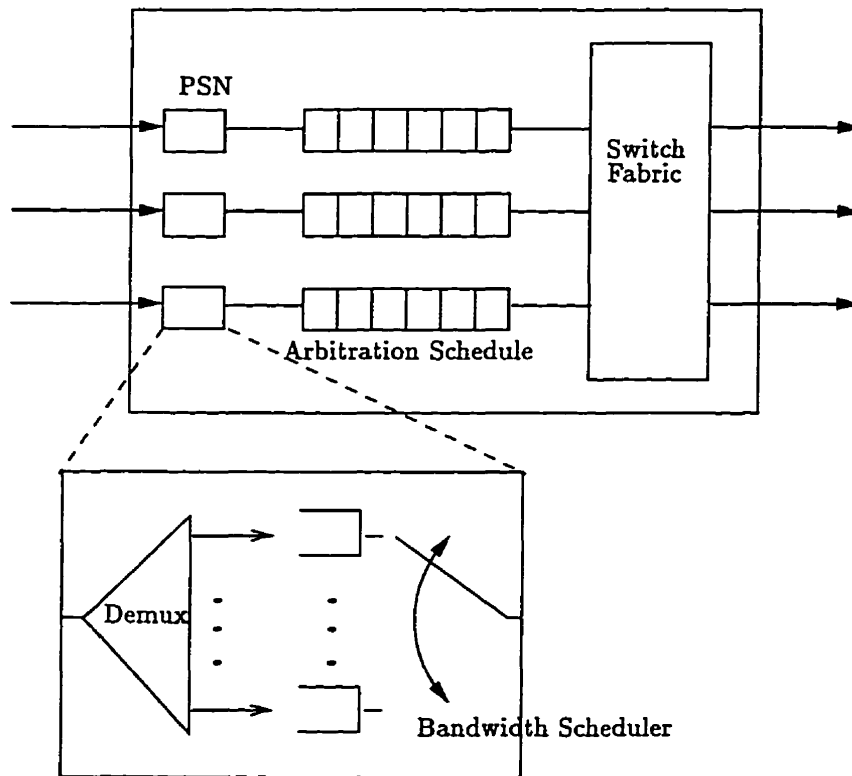


Figure 6.10: An input-buffered switch

condition for the resource allocation at output port  $j$  is

$$\sum_{i=1}^F M_{i,j} \leq f. \quad (6.7)$$

In addition, it is obvious that input link  $i$  cannot be overbooked; i.e.,

$$\sum_{j=1}^F M_{i,j} \leq f. \quad (6.8)$$

In other words, (6.7) is the constraint for resource allocation and (6.8) is a property of the connections in the input links.

One possible arbitration schedule for the previous example is given in Table 6.1. The table shows which output port each input port will be routed to. Note that it satisfies two required conditions:

1. There is at most one cell destined to each output port, and
2. There is a total of at least  $M_{i,j}$  cells from input port  $i$  to output port  $j$ .

These are the only two conditions needed to satisfy bandwidth guarantees in an input-buffered switch, and so there are many possible schedules that can do this. This schedule can be implemented with the Slepian-Duguid algorithm [1] [31], or an application of the Hungarian algorithm [32]. These algorithms are rather complex, however, and methods of improving on them are topics of further study.

Recall that the arbitration schedule dealt with guaranteeing bandwidth to input/output port pairs. A secondary scheduler is still needed to determine which queue that has the given input/output port pair will be served. In other words, this is a hierarchical scheduling scheme. The secondary scheduler is the scheduler inside the input port PSN. G. Kesidis in [35] proposed using idling HRR as the secondary scheduler, naming the scheme Guaranteed-Rate Round-Robin. In addition, semi-idling HRR can be used so

Slot	1	2	3	4	5	6
Input port 1	1	2	2	-	-	-
Input port 2	2	1	1	2	3	3
Input port 3	3	3	3	3	1	2

Table 6.1: Arbitration schedules

that the arbitration schedule as well as the HRR frame structure can be altered in the manner described in §6.1, while keeping the minimum-bandwidth property unchanged even when the arbitration schedule is changed. So, the minimum-bandwidth property result of the previous section is very useful here.

Service to best-effort traffic is more complex and is a topic of further study. The best-effort slots include slots that are not previously assigned; there are three such slots at input port 1 in Table 6.1. Best-effort slots are also created when an assigned slot is unused by an empty queue. Of the many schemes proposed in the literature for this purpose, one may be chosen; the AN2 switch uses a *parallel iterative matching* algorithm. However, recall that throughput is always lower for input-buffering, as opposed to output-buffering, when handling best-effort cells; so, input-buffering is a more attractive alternative if more non-best-effort traffic is used. In addition, there may be a bandwidth granularity problem with the round-robin arbitration frame. However, these disadvantages may be acceptable considering the lower memory bandwidth requirement.

The prototype AN2 switch was implemented. It was shown to work with  $c$  being 1 Gbps on a  $16 \times 16$  crossbar switch.



## Chapter 7

# Summary and Conclusions

### 7.1 Summary

This thesis has concentrated on a deterministic analysis of transmission and queueing in an ATM network. The issues and solutions discussed form a complete methodology for providing the two basic services that an integrated network needs; i.e., a method of providing reliable CBR service and reasonable throughput to simple best-effort traffic such as UBR.

The thesis focused on bandwidth scheduling in PSNs. Queueing is done on a per-VCC or per-VPC basis with the possible exception of best-effort traffic. The four design goals for bandwidth scheduling are:

1. Bandwidth guarantees,
2. Implementability at ATM speeds,
3. Efficiency in bandwidth usage, and
4. Good idle bandwidth distribution.

The notion of a bandwidth guarantee is indicated by the minimum-bandwidth property, which measures how well a scheduler can provide this guarantee, if at all. A point that was emphasized was that the idle bandwidth distribution has a specific purpose: to give service to connections that can make use of it; e.g., VBR connections and best-effort traffic. A design using a combination of idling and non-idling schedulers in a hierarchical scheduling scheme was proposed in order to apply idle bandwidth where it is needed.

The minimum-bandwidth property of a number of well-known schedulers were described. Several timestamping schedulers have very good minimum-bandwidth properties, but they are difficult to implement. The minimum-bandwidth property of idling HRR was found to be reasonably small if multilevel-assignment is used.

Moving from the single queue results to the case of an end-to-end virtual channel consisting of minimum-bandwidth PSNs, end-to-end delays were derived for the case where the source is and is not  $(\sigma, \rho)$ -constrained, although an absolute bound is only achieved in the latter case. These results were also extended to end-to-end connections consisting of virtual paths. One important feature of the minimum-bandwidth property is that it is an if and only if condition on absolute delay guarantees for  $(\sigma, \rho)$ -constrained sources. A similar criterion, called the linear service curve, is a sufficient but not a necessary condition for this. Buffer sizing results, assuming that the source is  $(\sigma, \rho)$ -constrained, can be achieved at each PSN queue along the virtual channel. The result shows that, in general, the buffer sizes increase linearly with the number of PSNs along the virtual channel. If this is an undesirable property, it can be avoided by using schedulers which can perform  $(\sigma, \rho)$  shaping on the departure process; for example, idling HRR.

The transmission problem for prerecorded video was given as an useful application for the previous results. With prerecorded video, the statistics of the connection are known before transmission, and a transmission scheme called piecewise-CBR was proposed. Piecewise-CBR can provide guaranteed service, while allowing best-effort traffic,

such as ABR, the use of reserved but unallocated bandwidth.

Finally, implementation issues were discussed, and multilevel semi-idling HRR was recommended as an implementable scheduler for high speeds because it has a good minimum-bandwidth property, active control of idle bandwidth and flexible bandwidth allocations. The use of output-buffering was recommended because it offers the best performance; however, it may not be implementable at high ATM speeds. A solution to this problem is to use input-buffered switches. With input-buffering, the input ports are the PSNs and arbitration across the switch fabric is required. The proposed arbitration scheme uses the Slepian-Duguid algorithm to guarantee bandwidth.

## 7.2 Conclusions and Future Directions

The use of deterministic bounds to guarantee performance is a crucial first step in designing an integrated network. CBR and best-effort are the core services that the network should provide. The Internet community recognizes this and is moving towards such a design with Resource Reservation Protocol (RSVP) [64] as an addition to the existing Internet. At this point, the research is mature enough that work should focus on implementing the theory in the field. This should be used as a starting point, and tentative conclusions can then be reached as to the usefulness of such a network. This is critically important in view of past experiences: no researcher could have envisioned the popularity of the Internet. By observing the experiences in the field, researchers can then better direct their future efforts. In particular, it would be more clear as to which of the following services should be emphasized: ABR, ABT and/or statistical multiplexing VBR.

How the ABT and ABR services can be realized is a topic of current and future study. If large loss can be tolerated because there is retransmission, it may be possible to just use the idle bandwidth to serve ABR traffic above the MCR; otherwise, temporary allocation

of bandwidth is likely needed. In general, these services can be built over the framework described in this thesis.

This thesis has made extensive use of deterministic bounds to provide QoS. Statistical bounds exist for schedulers but they are heavily dependent on factors such as the traffic load, traffic models and network topology. The use of statistical bounds may be possible in the future once more is known about such aspects of ATM networks through research and field experience. As an example, the delay bounds of this thesis, which are based on the minimum-bandwidth property, are expected to be larger than the typical delays. If statistical bounds for delay can be found which can be met with a very high probability — for example, a bound that can be met 99% of the time — then a notion of statistical guarantees can be formulated. Such methods require a very robust statistical model: one which would hold for a very general class of traffic sources and which would be valid for varying network topology and varying statistics of the interfering traffic.

### 7.3 Related Work: ATM via Satellite

In addition to the work for this thesis, I worked on multiple access schemes for ATM over a satellite network. The scenario that was considered uses a geostationary satellite where ground station terminals communicate with the satellite using ATM on the Ka frequency band.

The use of satellite has a number of significant advantages over terrestrial ATM: a satellite can cover a large area, is immune to terrestrial disasters, can offer broadband links and can be accessed simply, quickly and at a relatively low cost with very small terminals, without the infrastructure costs needed for ground networks using fiber optics links [50].

In [36], a simple, hybrid ATM-satellite protocol layer in which ATM service classes

are mapped to uplink access methods was proposed. The uplink access and resource allocation approaches based on this model were described in detail. The scheme can show how different QoS can be provided for by using a combination of the different access schemes. Scheduling for the uplink portion of the satellite network was also discussed in the paper. Again, the HRR scheduler was recommended for scheduling based on the grounds of performance, flexibility and implementability.



# Appendix A

## Glossary

### Terms

$(\sigma, \rho)$ constraint	A mathematical definition for the burstiness envelope of a process.
$(\sigma, \rho)$ shaping	Controlling the $(\sigma, \rho)$ constraint of the departure process of a queue in a manner such that it is independent of arrival process.
Bandwidth granularity	The difference in the bandwidth between two consecutive possible bandwidth allocations.
Best-effort traffic	Traffic that is to be served without permanent (over the lifetime of the connection) bandwidth guarantees, although it may be temporarily guaranteed bandwidth (e.g. bandwidth to ABR above the MCR)
Busy period	An interval of time in which a given queue is always nonempty.
Busy period of the reference queue:	An interval of time in which the fluid reference queue

	is always nonempty.
Connection	In the context of this thesis, a VCC.
Cell delay variation	The difference between the maximum and minimum delays experienced by the cells of a connection.
Cut-through	A mechanism in PSNs in which a cell may depart its queue at the same time that it arrives there.
Excess bandwidth	The total unallocated bandwidth $U$ .
Fluid	The method of modeling whereby cells are infinitely divisible and multiple cells can be in transmission at the same time.
GPS fairness	The ability of a scheduler to distribute the idle bandwidth bandwidth to queues in proportion to their allocated rates ( $\rho$ 's).
Hierarchical scheduling	A scheduling scheme whereby a scheduler may service logical queues which may contain a scheduler that serves other queues.
Idle bandwidth	The slots that are made available when they are unused by empty queues.
Idling scheduler	A scheduling scheme that does not necessarily serve any queue with bandwidth guarantees even when one or more of them is nonempty.
Leaky bucket	An algorithm to impose a $(\sigma, \rho)$ constraint on an arrival process to the network; also called the Generic Cell Rate Algorithm.
Linear service curve	The $(\sigma^{\text{svc}}, \rho)$ service curve.

Link	The interconnecting medium between nodes.
Minimum-bandwidth PSN:	A PSN that has a minimum-bandwidth property.
Node	A PSN, end-user or UPC device in the network.
Non-blocking switch fabric:	A switch fabric which can route a cell from each input port simultaneously given that they are all destined to different output ports.
Non-idling scheduler	A scheduling scheme that always serves a queue with a bandwidth guarantee when one or more of them is nonempty.
Overbooking	Where the allocated bandwidth is less than the required bandwidth.
Per-VCC queueing	Queueing whereby cells with different VCIs are put into different FIFO queues.
Per-VPC queueing	Queueing whereby all cells with the same VPI are put into their own single FIFO queue, and cells with different VPIs are put into different FIFO queues.
Policing	The process of controlling the traffic characteristics of the arrival process to the network.
Process	In the context of this thesis, an arrival or departure times process.
Processor sharing node	A component of the switch containing a set of queues and a scheduler.

Prerecorded video	A video trace that is entirely created and stored before transmission; also called stored video.
Semi-idling scheduler	A round-robin scheduling scheme which serves a queue in variable positions in a frame; an idling round-robin scheduler serves a queue in fixed positions.
Slot	The interval of time between two consecutive clock ticks (integer points) in the real time line. Also, the amount of time required to transmit one cell on a link which is equal to $c$ cells/sec.
Speed-up	The technique of decreasing the duration of a slot in a switch so that the overall rate of a bus inside the switch is faster than the input links.
Service curve	A mathematical definition for the minimum service in a busy period of a PSN queue.
Traffic descriptors	Parameters used to describe some characteristics that a source must comply with.
User Network Interface	The interface between an end-user and the network.
Virtual channel	Tandem nodes comprising a path on which data is sent, also called a virtual circuit.
Virtual channel connection:	A single user connection passing through a number of nodes.
Virtual path connection	A group of individual VC bundled together passing through a number of nodes.
Virtual time function	The function $v(\cdot)$ used in calculating the VFTs.

**Underbooking**                Where the allocated bandwidth is greater than the required bandwidth.

**Work-conserving scheduler:** Same as the non-idling scheduler.

**Work**                        The number of cells served; also referred to as throughput.

## Notation

The following is a list of all symbols that have global scope in the thesis. Unless otherwise stated, all time are in units of slots, all queue sizes are in units of cells and all bandwidths are in units of cells/slot.

$(s, t]$	The (continuous) interval from $s$ to $t$ including $t$ but not including $s$ ; an empty interval if $s = t$ .
$(x)^+$	Equivalent to $\max(x, 0)$ .
$t^-$	The instant just before the departure of a cell at time $t \in \mathbb{Z}^+$ .
$O(f(x))$	Meaning on the order of $f(x)$ .
$\lceil x \rceil$	The smallest integer greater than or equal to $x$ .
$\lfloor x \rfloor$	The largest integer smaller than or equal to $x$ .
$\mathbb{Z}^+$	The set $\{0, 1, 2, \dots\}$ .
$\mathbb{R}^+$	The interval $[0, \infty)$ .
$\delta_i$	The end-to-end delay of cell $i$ .
$\mu$	The minimum-bandwidth property.

$\mu_{\text{eff}}$	The effective minimum-bandwidth of the queueing system.
$\mu_{i,j}$	The effective minimum-bandwidth property of nodes $i, i + 1, \dots, j$ .
$\Pi$	The total propagation delay of a virtual channel.
$\pi_k$	The propagation delay from node $k$ to the next node.
$\phi_w$	The service bandwidth in the network during window $w$ for the video connection.
$\bar{\phi}$	The time average of the bandwidth allotments of the prerecorded video connection.
$\phi_{\min}$	The minimum service bandwidth allotment of the prerecorded video connection.
$\phi_{\max}$	The maximum service bandwidth allotment of the prerecorded video connection.
$\vec{\phi}$	The traffic descriptor describing the bandwidth allocations and their durations; i.e., $\vec{\phi} = \{\{\phi_1, l_1\}, \{\phi_2, l_2\}, \dots, \{\phi_w, l_w\}\}$ .
$\rho^{\text{req}}, \rho_n^{\text{req}}$	The bandwidth requirement of the queue under consideration (queue $n$ ).
$\rho, \rho_n$	The bandwidth allocation of the queue under consideration (queue $n$ ).
$\rho_*$	The sum of the $\rho$ 's of the sources entering a VPC, other than the connection under consideration.
$\sigma_*$	The sum of the $\sigma$ 's of the sources entering a VPC, other than the connection under consideration.
$\tau$	The duration of a video frame; e.g., 1/24 sec.

$\varphi_l$	The bandwidth allocation at level $l$ of the HRR frame.
$\xi_l$	A value that satisfies $ K_l(s, t) - (t - s)\varphi_l  \leq \xi_l$ .
$\xi$	Equivalent to $\sum_{l=1}^L \xi_l$ .
$A(s, t)$	If $s \leq t$ , the total number of arrivals in the interval $(s, t]$ ; if $s > t$ , equivalent to $-A(t, s)$ .
$a_i, a_i^n$	Arrival time of cell $i$ at the queue under consideration (queue $n$ ) in a PSN.
$a_i(k)$	Arrival time of cell $i$ at node $k$ along the virtual channel.
$B_{\mathcal{P}}$	The size of the playback buffer for video transmission in the fluid reference network.
$B_{\mathcal{P}}$	The size of the playback buffer for video transmission in the actual virtual channel.
$B_k$	The buffer requirement (to ensure no overflow) at the PSN $k$ along the virtual channel.
$c$	The bandwidth of the output links in units of cells/sec.
$c_i^n$	Cell $i$ at queue $n$ in a PSN.
$D_{\mathcal{P}}$	The initial playback delay for prerecorded video transmission in the fluid reference network.
$D_{\mathcal{P}}$	The initial playback delay for prerecorded video transmission in the actual virtual channel.
$D(s, t)$	The total number of departures in the interval $(s, t]$ .

$d_i, d_i^n$	Departure time of cell $i$ from the queue under consideration (queue $n$ ) in a PSN — the arrival times are $\{a_i\}$ or $\{a_i^n\}$ .
$d_i(k)$	Departure time of cell $i$ from the $k^{\text{th}}$ node along the virtual channel — the arrival times are $\{a_i(k)\}$ .
$E_i^n$	The timestamp of cell $i$ at queue $n$ in a PSN with the HOL scheduler.
$E^n(t)$	The timestamp of the HOL cell at queue $n$ at time $t$ in a PSN with the HOL scheduler.
$\mathcal{F}_i, \mathcal{F}_i^n$	Departure time of cell $i$ from an isolated fluid reference queue with arrivals $\{a_i\}$ or $\{a_i^n\}$ and a service rate of exactly $\rho$ or $\rho_n$ cells/sec; also called the VC-VFT.
$\mathcal{F}_i(k)$	Departure time of cell $i$ from the fluid queue given that the arrivals are $\{a_i(k)\}$ .
$\mathcal{F}_i^a$	The VirtualClock-Virtual Finishing Time (VC-VFT) of the cell $i$ in the aggregate arrival process to the VPC queue under consideration.
$F_i, F_i^n$	The VFT of cell $i$ in the queue under consideration (queue $n$ ).
$F$	The number of input or output ports in the ATM switch; i.e., the switch is $F \times F$ .
$f$	The size of the WRR frame.
$f_l$	The size of the level $l$ HRR frame.
$I(i)$	The index of the $i^{\text{th}}$ cell within a VPC.
$K$	The number of nodes in the virtual channel, excluding the UPC device.

$k_l$	The number of slots allocated to the connection in the level $l$ HRR frame.
$l_w$	The duration of window $w$ in the prerecorded video connection.
$L$	The number of levels in the HRR frame structure.
$N$	Number of queues with bandwidth guarantees in the PSN.
$K_l(s, t)$	The level $l$ HRR service opportunities in the interval $(s, t]$ .
$K(s, t)$	The total HRR reserved slots in the interval $(s, t]$ ; i.e., $K(s, t) = \sum_{l=1}^L K_l(s, t)$ .
$n_l$	The number of slots allocated to level $l + 1$ in a level $l$ HRR frame.
$P$	The index of the node containing the playback buffer; i.e., $P = K + 1$ .
$\mathcal{Q}(t)$	The occupancy of the fluid reference queue at time $t$ .
$Q(t)$	The occupancy of a PSN queue at time $t$ .
$R(t)$	The total number of cells required by the prerecorded video decoder at time $t$ after playback initiates.
$\mathcal{S}(t)$	The total number of cells received at the destination node in the reference network in $[0, t]$ .
$S(t)$	The total number of cells received at the destination node in the arbitrary virtual channel in $[0, t]$ .
$T$	The total duration of the prerecorded video playback.
$U$	The total excess (unallocated) bandwidth in the PSN.
$W$	The total number of windows in the video connection.



# Bibliography

- [1] T.E. Anderson, S.S. Owicki, J.B. Saxe, and C.P. Thacker. High speed switch scheduling for local area networks. Technical report, Digital Systems Research Center, April 26, 1993.
- [2] S. Asmussen. *Applied probability and queues*. Wiley, Chichester West Sussex, 1987.
- [3] R.Y. Awdeh and H.T. Mouftah. Survey of ATM switch architectures. *Computer Networks and ISDN Systems*, Vol. 27:1567–1613, 1995.
- [4] J.C.R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. In *Proc. ACM SIGCOMM*, pages 143–156, 1996.
- [5] J.C.R. Bennett and H. Zhang. WF<sup>2</sup>Q: Worst-case weighted fair queueing. In *Proc. IEEE INFOCOM*, pages 120–128, 1996.
- [6] F. Bonomi and K.W. Fendick. The rate-based flow control framework for the available bit rate ATM service. *IEEE Network*, Vol. 9, No. 2:25–39, 1995.
- [7] P.E. Boyer and D.P. Tranchier. A reservation principle with applications in the ATM traffic control. *Computer Networks and ISDN Systems*, Vol. 24:321–334, 1996.
- [8] H.J. Chao. A novel architecture for queue management in the ATM network. *IEEE JSAC*, Vol. 9, No. 7:1110–1118, 1991.

- [9] R. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Trans. Inform. Theory*, Vol. 37:132–141, Jan. 1991.
- [10] R. L. Cruz. A calculus for network delay, Part 1: Network elements in isolation. *IEEE Trans. Inform. Theory*, Vol. 37:114–131, 1991.
- [11] R.L. Cruz. Service burstiness and dynamic burstiness measures: A framework. *Journal of High Speed Networks*, Vol. 1, No. 2:105–127, 1992.
- [12] R.L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE JSAC*, Vol. 13, No. 6:1048–1056, 1995.
- [13] M. de Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN*. 2nd Ed., Ellis Horwood, Toronto, 1993.
- [14] G. de Veciana and G. Kesidis. Bandwidth allocation for multiple qualities of service using generalized processor sharing. *IEEE Trans. Info. Theory*, Vol. 42, No. 1:268–271, 1996.
- [15] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, pages 3–26, Oct. 1990.
- [16] N.R. Figueira and J. Pasquale. An upper bound on delay for the VirtualClock service discipline. *IEEE/ACM Trans. Networking*, Vol. 3, No. 4:399–408, 1995.
- [17] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Trans. Networking*, Vol. 3, No. 4:365–386, 1995.
- [18] ATM Forum. *ATM User-Network-Interface Specification Version 3.1*. Wiley and Sons., Toronto, 1995.

- [19] ATM Forum. Draft version 3.0 of ATM forum traffic management specification version 4.0. distributed by FORE Systems, Inc., April 15, 1995.
- [20] D. Le Gall. MPEG: A video compression standard for multimedia applications. *Commun. of the ACM*, Vol. 34, No. 4:47–58, 1991.
- [21] M.W. Garret and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *Proc. ACM SIGCOMM*, pages 269–280, 1994.
- [22] L. Georgiadis, R. Guérin, V. Peris, and K.N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE Trans. Networking*, Vol. 4, No. 4:482–501, Aug. 1996.
- [23] S.J. Golestani. A framing strategy for congestion management. *IEEE JSAC*, Vol. 9, No. 7:1064–1077, 1987.
- [24] S.J. Golestani. A self-clocked fair queueing scheme for broadband applications. *Proc. IEEE Infocom*, Vol. 2:636–646, 1994.
- [25] P. Goyal, S. Lam, and H. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 273–284, Durham, NH, 1995.
- [26] P. Goyal, S. Lam, and H. Vin. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. In *Proc. ACM SIGCOMM*, pages 157–168, Stanford, CA, Aug. 1996.
- [27] A.G. Greenberg and N. Madras. How fair is fair queueing? *Journal of the ACM*, Vol. 39, No. 3:568–598, 1992.
- [28] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A simple and efficient service for multiple time-scale traffic. In *Proc. ACM SIGCOMM*, pages 219–230, 1995.

- [29] R. Handel, M. N. Huber, and S. Schroder. *ATM Networks: Concepts, Protocols, Applications*. Addison-Wesley, Don Mills, Ontario, 1994.
- [30] E. Horowitz and S. Sahni. *Data Structures in Pascal 3rd Ed.* Computer Science Press, Toronto, 1990.
- [31] J.Y. Hui. *Switching and Traffic Theory for Integrated Broadband Networks*. Kluwer Acad. Publ., Boston, 1990.
- [32] A. Hung, P. Coutu, and M. Coté. A nonblocking assignment algorithm for input-buffered switches. Technical Report 96-14, Univ. of Waterloo, Jan. 27, 1997.
- [33] A. Hung and G. Kesidis. Bandwidth scheduling for wide-area ATM networks using virtual finishing times. *IEEE/ACM Trans. Networking*, Vol. 4, No. 1:49–54, 1996.
- [34] A. Hung and G. Kesidis. Performance evaluation of multilevel-assignment hierarchical round-robin bandwidth scheduling for ATM. In *15th International Teletraffic Congress*, pages 1247–1256, Washington, D.C., 1997.
- [35] A. Hung and G. Kesidis. Guaranteed-rate round robin scheduling for input-buffered ATM switches. Technical Report 96-2, Univ. of Waterloo, Feb. 1, 1997.
- [36] A. Hung, M.-J. Montpetit, and G. Kesidis. ATM via satellite: A framework and implementation. To appear in *ACM-Baltzer Journal of Wireless Networks* special issue on *Hybrid and Satellite Comm. Networks*, 1997.
- [37] T. Ibaraki and T. Kameda. Multi-frame isochronous service for ATM networks: Stop-and-Go revisited. preprint, 1997.
- [38] Texas Instruments. <http://www.ti.com/sc/docs/memory/guide.htm>.
- [39] ITU. Traffic control and congestion control in B-ISDN. Technical Report I.371, 1995.

- [40] C.R. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *Proc. IEEE Globecom*, pages 300.3.1–300.3.9, 1990.
- [41] G. Karlsson. Asynchronous transfer of video. *IEEE Communications Magazine*, Vol. 34, No. 8:118–126, 1996.
- [42] M.J. Karol, M.G. Hluchyj, and S.P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Trans. Commun.*, Vol. 35, No. 12:1347–1356, 1987.
- [43] S. Keshav. On the efficient implementation of fair queueing. *Internetworking: Research and Experience*, Vol. 2:157–173, 1991.
- [44] G. Kesidis. *ATM Network Performance*. Kluwer Academic Publishers, Boston, MA, 1996.
- [45] S.S. Lam, S. Chow, and D.K.Y. Yau. An algorithm for lossless smoothing of MPEG video. In *Proc. ACM SIGCOMM*, pages 281–293, 1994.
- [46] K. Lee. Performance bounds in communication networks with variable-rate links. In *Proc. ACM SIGCOMM*, pages 126–136, 1995.
- [47] N. McKeown. *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, U. C. Berkeley, 1995.
- [48] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input queued switch. In *Proc. IEEE INFOCOM*, pages 296–302, 1996.
- [49] J.M. McManus and K.W. Ross. Video on demand over ATM: Constant-rate transmission and transport. In *Proc. IEEE INFOCOM*, pages 1357–1362, 1996.

- [50] M.-J. Montpetit and A. Hung. Design issues for ATM over satellite. In *18th Biennial Symposium on Communications*, pages 83–86, Kingston, Ont., Canada, 1996.
- [51] P. Pancha and M. El Zarki. MPEG coded video and BISDN. *IEEE Communications Magazine*, Vol. 32, No. 5:54–66, 1994.
- [52] P. Pancha and M. El Zarki. Leaky bucket access control for VBR MPEG video. In *Proc. IEEE INFOCOM*, pages 796–803, 1995.
- [53] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control: The multiple node case. *IEEE/ACM Trans. Networking*, Vol. 2, No. 2:137–150, 1994.
- [54] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Trans. Networking*, Vol. 1, No. 3:344–357, Jun. 1993.
- [55] J-F Ren, J.W. Mark, and J.W. Wong. A dynamic priority queueing approach to traffic regulation in scheduling in B-ISDN. In *Proc. IEEE Globecom*, pages 612–618, 1994.
- [56] J. Rexford, A. Greenberg, and F. Bonomi. Hardware-efficient fair queueing architectures for high-speed networks. In *Proc. IEEE INFOCOM*, pages 638–646, 1996.
- [57] J.W. Roberts. Virtual spacing for flexible traffic control. *International Journal of Communication Systems*, Vol. 7:307–318, 1994.
- [58] J.W. Roberts. What ATM transfer capabilities for the B-ISDN? In *First Workshop on ATM Traffic Management*, pages 181–194, 1995.

- [59] J. Salehi, Z.-L. Zhang, J. Kurose, and D. Towsley. Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing. In *Proc. ACM SIGMETRICS*, pages 115–120, 1996.
- [60] F.A. Tobagi. Fast packet switch architectures for broadband integrated services digital networks. *Proc. of the IEEE*, Vol. 78, No. 1:133–167, 1990.
- [61] G. De Veciana, G. Kesidis, and J. Walrand. Resource management in wide-area ATM networks using effective bandwidths. *IEEE JSAC*, Vol. 13, No. 6:1081–1090, 1995.
- [62] H. Vin. Talk on fairness in scheduling. In *Workshop on Packet Scheduling Algorithms*, XEROX Parc, San Jose, CA, 1996.
- [63] M. Vishnu. *Design of a Versatile ATM Switching Node with a Per-VC Architecture*. PhD thesis, Univ. of Waterloo, 1996.
- [64] P.P. White. RSVP and integrated services in the Internet: A tutorial. *IEEE Communications Magazine*, Vol. 35, No. 5:100–106, 1996.
- [65] G.G. Xie and S.S. Lam. Delay guarantee of the Virtual Clock server. *IEEE/ACM Trans. Networking*, Vol. 3, No. 6:683–689, 1995.
- [66] D. Yates, J. Kurose, D. Towsley, and M.G. Hluchyj. On per-session end-to-end delay distributions and the call admission problem for real-time applications with QOS requirements. In *Proc. ACM SIGCOMM*, pages 2–12, 1993.
- [67] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, Vol. 83, No. 10:374–396, 1995.
- [68] L. Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. In *Proc. ACM SIGCOMM*, pages 19–29, Sept. 1990.