A STUDY OF MULTIPLEXING ON TO A VARIABLE-BIT RATE OUTPUT CHANNEL IN INTEGRATED-SERVICE NETWORKS

BY

Chan-weng Lai

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF PHILOSOPHY

DIVISION OF INFORMATION ENGINEERING

THE CHINESE UNIVERSITY OF HONG KONG

AUGUST 1995

Acknowledgement

I am grateful to Dr. S. C. Liew for suggesting this topic and supervising this research. I also thank Prof. Tony Lee and Dr. W. S. Wong for their constructive opinions.

Abstract

This thesis investigates *soft multiplexing*, an issue that arises in VP-based ATM networks. The main distinguishing feature of this form of multiplexing (as opposed to traditional statistical multiplexing) is that the output channel is a *variable-bit rate* logical channel. We assume that the overall multiplexed output traffic must conform to the (δ, ρ) *specification.* The issue in soft multiplexing is how to schedule the transmission of traffic arriving from different sessions using the two resources: ρ, the transmission capacity, and δ, the power of saving up transmission capacity. To motivate this study, we show by simulations that the use of (δ, ρ) VP's in ATM networks can yield better statistical performance (the delay/backlog distribution) than the use of deterministic VP's can. We compare three soft multiplexing schemes, showing the trade-off between performance guarantee to individual sessions and the average performance of all sessions being multiplexed. In particular, we analyze in detail one of the soft multiplexing schemes proposed, called the *rate proportional token passing (RPTP).* We show that when the sources are controlled by leaky buckets before they are multiplexed, a RPTP multiplexer can provide deterministic delay and backlog guarantee. Two implementations of RPTP are studied and compared.

Contents

1 Introduction **1**

1.1 Where May Soft Multiplexing Occur? 2

1.1.1 Multiplexing VC's on to a VP 2

1.1.2 Virtual Private Networks 5

1.2 Survey of Previously Proposed Hard Multiplexing Schemes . . . 7

1.3 Contributions of This Thesis 8

1.4 Organization of This Thesis 10

2 Effect of (δ, ρ) Channels in ATM Networks **12**

2.1 Leaky Bucket 13

2.2 (δ, ρ) Channel 14

2.3 Comparison of Deterministic VP's and (δ, ρ) VP's 17

2.4 A Simulation Study : The Effect of δ 20

2.5 Summary of This Chapter 23

3 Soft-Multiplexing Scheduling Schemes **26**

3.1 Issues in Soft Multiplexing 27

3.2 First Come First Serve (FCFS) 30

3.3 Fixed-resource Allocation 32

3.4 Excess Token Passing 35
3.5 Simulation Results 38
3.6 Summary of This Chapter 42

4 Analysis of Rate Proportional Token Passing **44**
4.1 The Fictitious System 45
4.2 Leaky-Bucket-Controlled Sources 49
4.3 Delay Bound for All Work Conserving Soft Multiplexers 51
4.4 The All-Greedy Bound in a RPTP Multiplexer 53
4.5 Calculation of the Worst-Case Delay in a RPTP Multiplexer . . 56
4.6 Summary of This Chapter 61

5 Implementation of RPTP **63**
5.1 Virtual Time Implementation 64
5.2 Leaky Bucket Implementation 70
5.3 Summary of This Chapter 72

6 Conclusion **73**

A End-to-end Delay/Backlog Bound in ATM Networks **76**

Bibliography **81**

Chapter 1

Introduction

One common assumption of traditional multiplexing schemes is that the output channel is constant-bit rate. This is the case, for example, when traffic is multiplexed on to a physical link. However, this may not always be the case, for example, in Asynchronous Transfer Mode (ATM) networks in which virtual channels (VC's) are multiplexed on to virtual paths (VP's) which are also logical channels themselves [12, 13]. The capacity of a logical channel is not hard limited and can be varied depending on the policing entity utilized. Because of this variable-bit rate (VBR) characteristic of the output channel, most existing multiplexing schemes are not applicable.

This thesis is devoted to the investigation of multiplexing with variable-bit rate output channel, which we refer to as *Soft Multiplexing*. We assume that, in soft multiplexing, the traffic on the output channel must conform to the (δ, ρ) specification , or in other words, the output channel is a (δ, ρ) channel. Multiplexing with constant-bit rate (CBR) output channel will be called *Hard Multiplexing*.

The rest of this chapter is organized as follows. For motivations, Section 1.1 gives two scenarios where soft multiplexing may occur. To put things in context, Section 1.2 gives a brief review of hard-multiplexing schemes. After that, the contributions and organization of this thesis are given in Section 1.3 and Section 1.4 respectively.

1.1 Where May Soft Multiplexing Occur?

For motivations, we show in this section two scenarios where soft multiplexing (i.e. multiplexing on to a (δ, ρ) channel) may occur. The first scenario is when VC's are being multiplexed on to a VP inside an ATM network. The different VC's belong to different end users or network-service subscribers, and the multiplexing is being performed by the network operator to optimize network usage. The second scenario is when a virtual private network (VPN) is set up using VP's in the ATM networks. The VPN belongs to one network-service subscriber (e.g. a private corporation leasing the communication facilities from the network operator) and the multiplexing is performed by the subscriber outside the ATM network.

1.1.1 Multiplexing VC's on to a VP

The Asynchronous Transfer Mode (ATM) has been accepted as the mechanism for transporting information in Broadband Integrated Services Digital Networks (B-ISDN). Optical transmission links with transmission rate in the range of Mb/s to Gb/s are expected to be used extensively in these networks. Many services (such as audio and video), on the other hand, will only need from several kb/s to

a few Mb/s of bandwidth. If a virtual circuit is used to carry each information stream, a physical link will then contain a huge number of simultaneous virtual circuits. It is prohibitive to have to monitor, control, and manage the traffic flows of thousands of virtual circuits on each link. Partially as a means to tackle this problem, the idea of two-level virtual-connection hierarchy consisting of virtual channels (VC) and virtual paths (VP) has been conceived [12, 13].

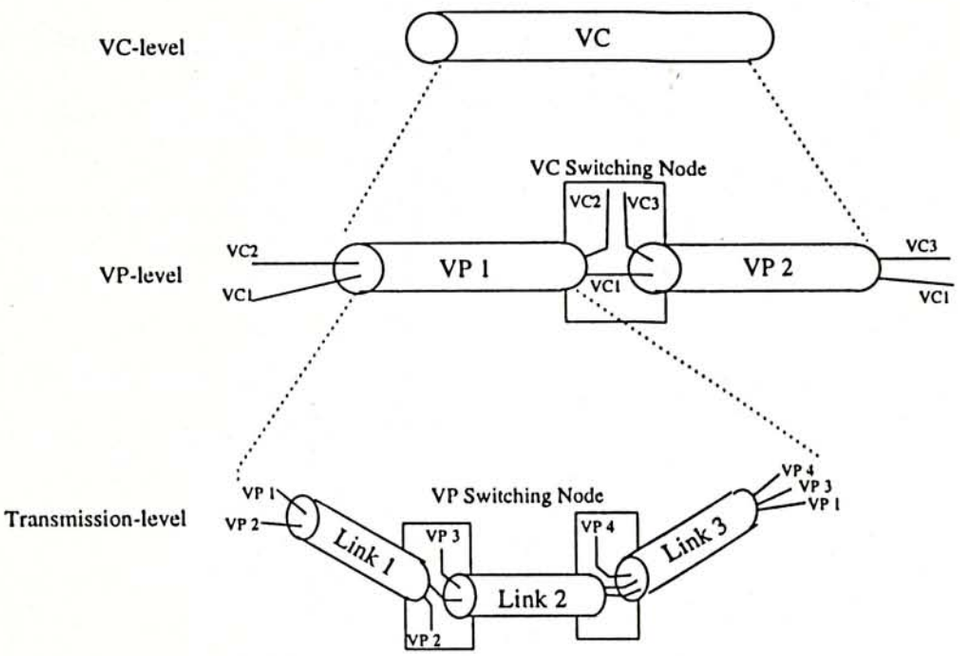

Figure 1.1: B-ISDN transport architecture.

An ATM-based B-ISDN transport architecture is shown in Fig. 1.1. A VP is a collection of VC's to be switched as a unit. The resulting network is structured into a hierarchy of three levels: VC, VP, and transmission levels [12, 13]. The VC-level network, logically, supports end-to-end connections. The VP-level network interconnects VC switching nodes and routing of VC's is based on the VP-level network. At VC switching nodes, VC's are multiplexed on to and demultiplexed from VP's. Note that both VC-level network and VP-level network

are logical. The transmission-level network, which is the physical network, is provided by SDH or other existing transmission system. Routing of VP's is based on the transmission-level network and, in VP switching nodes, VP's are multiplexed on to transmission links without the VC-level routing and control information being examined. The merits of using VP are:

1. Simplified network architecture: By managing the network at the VP level, the VC level can concentrate on service handling. This increases the network operatability and reliability.

2. Increased rearrangeability of the network: Network architecture can be rearranged to resolve network congestion by reconfiguring the VP network.

3. Reduction of the processing in VP switching nodes: Since a VP is a collection of VC's to be transported and switched as a unit, processing at VP switching nodes, e.g. call-by-call connection establishment, routing, bandwidth allocation and routing table renewal, can be reduced.

As illustrated in Fig.1.2, this three-layer arrangement presents a new multiplexing problem at the VP level. Specifically, unlike a physical link, a VP as a logical channel does not have a hard limit on its capacity, and if it behooves the network to do so, its capacity can be varied. In this thesis, we consider each VP to be a (δ, ρ) channel. One major characteristic of this kind of logical channels is that it allows certain level of bursty transmission. In Chapter 2, we describe the characteristics of (δ, ρ) channels in detail. We show that, the use of (δ, ρ) VP's can yield better statistical performance (delay/backlog distributions) in ATM networks than the use of CBR VP's can.

Because of the VBR nature of a (δ, ρ) channel, most existing multiplexing schemes cannot be applied directly to multiplex VC's on to the VP. As each VC may have different traffic characteristics and QOS requirements, one major issue is how to distribute the resources of output (δ, ρ) channel to the VC's such that all the requirements can be satisfied.

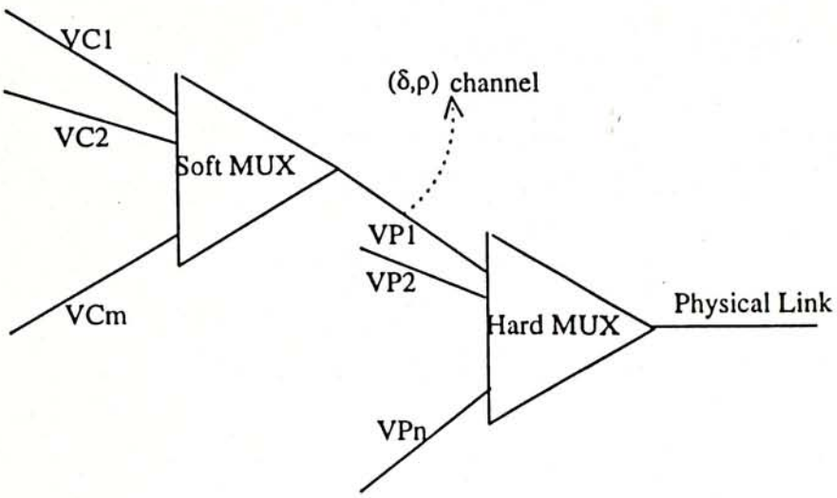

Figure 1.2: 2-layer multiplexing in B-ISDN.

1.1.2 Virtual Private Networks

One essential service to be provided by the future public network is the Virtual Private Network(VPN). For explanation, let us consider a hypothetical ATM public network in a city. Suppose that a bank in this city has several branches and users at these branches need to communicate with each other via various media, such as data, voice and video every day. What the public network provider can offer to this client is a VPN as illustrated in Fig. 1.3. A VPN consists of network resources such as user-network interfaces (UNI's) and permanent ATM

virtual connections (PVC's). A PVC is actually a dedicated VP with bandwidth controlled by a policing unit at the UNI's. The most common policing mechanism is the so-called *leaky bucket* [15, 16, 17]. The operation of leaky buckets are described in detail in Chapter 2. Note that a logical channels with its usage controlled by a leaky bucket is a (δ, ρ) channels. Therefore, we assume in this thesis that the PVC's are also (δ, ρ) channels.

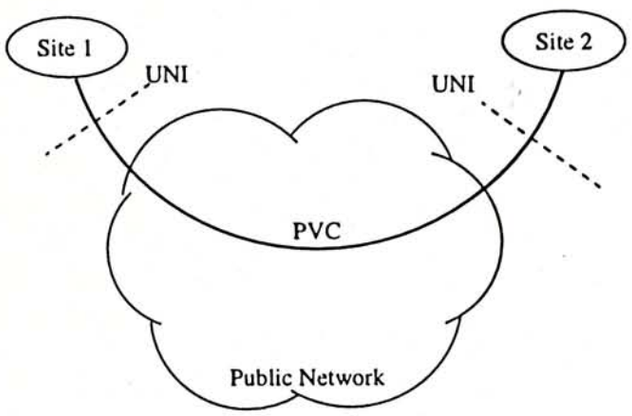

Figure 1.3: Virtual Private Network.

After setting up the VPN, the customer may multiplex various sessions on to the PVC, as depicted in Fig. 1.4. Therefore these sessions are being soft-multiplexed on to a (δ, ρ) channel. As each session may have its own characteristics and requirements, one major issue is how to distribute the allocated PVC transmission resources to each session and guarantee the performance of each session. Another important concern of the customer is how to make use of the VPN efficiently since it had already been paid for.

Note that in the first scenario in the previous subsection, multiplexing of VC's on to the VP's occurs inside the ATM networks and is performed by the network provider. The multiplexing in a VPN occurs outside the "backbone" network of the network provider and is the responsibility of the VPN subscriber.

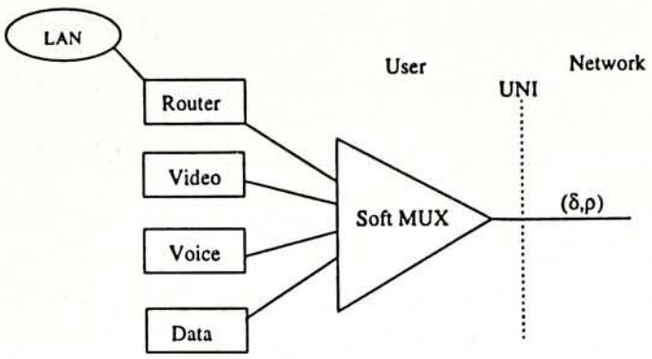

Figure 1.4: Soft Multiplexing on to a PVC.

1.2 Survey of Previously Proposed Hard Multiplexing Schemes

A challenge of multiplexing in broadband networks is to guarantee the performance of individual streams without sacrificing the efficiency achieved by statistical multiplexing. Several multiplexing schemes have been proposed in the past few years. Round-robin service discipline [2] intends to maintain fairness among sessions by picking one packet for service from each busy session in a cyclic way. Limitations of round robin is the assumptions of fixed-size packets and equal service share for all sessions. Later, Demers, et al, [3] removed these limitations and proposed a scheme called fluid-flow fair queueing (FFQ) based on idealized fluid-flow traffic which is infinitely divisible. Accordingly, sessions can be served in parallel and service can be distributed to busy sessions in proportion to their service shares at any time. An implementation of FFQ is packet-by-packet fair queueing (PFQ) which tries to serve the packets in the order they depart the idealized system. Parehk and Gallagar [4, 5] proved that both FFQ and PFQ (what they call Generalized Processor Sharing and Packet-by-packet

Generalized Processor Sharing) can provide end-to-end delay and backlog guarantees to individual sessions when combined with leaky bucket access control at the network boundary. They also proposed an implementation of PFQ using the concept of virtual time even though, later, Golestani [6] criticize that the dependence of PFQ to FFQ may lead to high computational complexity. By modifying the virtual time implementation of PFQ, Golestani propose a scheme called Self-clocked Fair Queueing(SCFQ) which depends on the actual queueing system only. And he showed that under this scheme, fairness among sessions can still be maintained. Other schemes include VirtualClock by Zhang [7] and Stop-and-Go Queueing [8, 9, 10] by Golestani.

1.3 Contributions of This Thesis

One common assumption of the multiplexing schemes in the preceding section is that the output channel is a physical channel whose bit rate is fixed. This thesis concerns multiplexing data on to a VBR channel. In particular, we focus on an output VBR channel with (δ, ρ) specification. The hard-multiplexing schemes in the preceding section cannot be applied directly in this case. The main contributions and results of this study are summarized below :

(1) A CBR channel can be considered as a (δ, ρ) channel with $\delta = 0$. To motivate the study of soft multiplexing, we have investigated the effect of a non-zero δ (the main distinguishing feature of soft multiplexing). In particular, from the network level point-of-view, we show by simulations that (δ, ρ) VP's can yield better statistical performance (delay/backlog distribution) in ATM networks than deterministic VP's (i.e. constant-bit rate

VP's) can. Assuming the FCFS scheduling scheme, this simulation, which can be found in Chapter 2, focuses on the delay and backlog distributions of the aggregate traffic of all VC's being multiplexed on to the same physical link.

(2) The performance (i.e. QOS) of individual sessions is important from the network-user's standpoint; and it depends on the scheduling scheme used. In Chapter 3, we construct and compare several soft-multiplexing scheduling schemes, namely first come first serve (which allows all sessions to share a common pool of transmission resources without explicit guarantee to individual sessions), fixed-resource allocation (which assigns dedicated transmission resources to individual sessions) and excess token passing (which allows a certain level of resource sharing while providing a certain degree of performance guarantee to each session). These comparisons show that there is a trade-off between overall average performance of all sessions being multiplexed and the hence performance guarantee to individual sessions. Our simulation results show that excess token passing can yield high output channel utilization, fairness among sessions and low loss/delay in the soft multiplexer.

(3) Besides the statistical performance, whether a soft multiplexer can provide worst-case guarantee (such as a bound on delay) is also an essential issue for it to be used in integrated-service networks. Chapter 4 analyzes the worst-case delay/backlog for one of many excess token passing schemes called the rate proportional token passing (RPTP). A fictitious system, which provides a mapping between soft-multiplexing and hard-multiplexing

schemes, is constructed. Via this fictitious system, we find that a lot of issues in soft multiplexing can be understood by borrowing results from previous works on hard multiplexing. For example, the hard-multiplexing scheme in the corresponding fictitious system of RPTP is actually fluid-flow fair queueing(FFQ). This allows us to derive the results of the RPTP system from Parekh's work. In particular, we show that when the traffic streams are controlled by leaky buckets before they are multiplexed, the delay and backlog for an arbitrary session in a RPTP multiplexer are bounded. The scenario under which this maximum delay/backlog occurs is derived.

(4) To investigate the feasibility and practicality, we consider and discuss two possible implementations of RPTP, namely virtual time implementation and leaky bucket implementation. Moreover, we have shown how previous implementations for hard multiplexer can be modified for soft-multiplexer implementation.

1.4 Organization of This Thesis

The rest of this thesis is organized as follow. In Chapter 2, we first explain in detail the characteristics of (δ, ρ) channels. Then, to motivate the study of soft multiplexing, we compare through simulations, the statistical performance (delay/backlog distribution) of ATM networks under the situations of using deterministic VP's and (δ, ρ) VP's. Chapter 3 discusses the major issues in soft multiplexing and introduces several soft-multiplexing schemes. The trade-off between performance guarantee to individual sessions and the overall average performance of all sessions is discussed and justified by simulation results. The

concepts of work conservation and greediness are introduced to classify and study soft multiplexing schemes systematically. Chapter 4 focuses on one of the soft-multiplexing schemes proposed, called the rate proportional token passing (RPTP). We prove that when the sources are controlled by leaky buckets, the delay and backlog in a RPTP multiplexer are bounded. Issues in finding the worst case delay are also discussed. In Chapter 5, we propose two possible implementations of RPTP. Some results of the end-to-end delay/backlog bound in ATM networks are given in the appendix.

Chapter 2

Effect of (δ, ρ) Channels in ATM Networks

An important issue in ATM networks is how should the transmission bandwidth of a VP be specified. One possibility is to fix the bandwidth of the VP's, or in other words, to use deterministic VP's (CBR VP's). However, as there is no bandwidth sharing among the deterministic VP's, statistical performance of the network may be low. In this chapter, we consider the use of (δ, ρ) VP's in ATM networks and show by simulations that (δ, ρ) VP's can yield better end-to-end statistical performance than deterministic VP's can. Assuming the FCFS scheduling scheme, our simulations focus on the overall delay and backlog distributions of the aggregate traffic of all sessions being multiplexed on to the same physical link.

In Section 2.1, we first describe the concept of the leaky bucket, which is closely related to (δ, ρ) channels. After that, the characteristics of (δ, ρ) channels are described and compared to the characteristics of physical channels in Section

2.2. Section 2.3 discusses the pros and cons of (δ, ρ) VP's and deterministic VP's. Finally, our arguments are supported by simulation results in Section 2.4.

2.1 Leaky Bucket

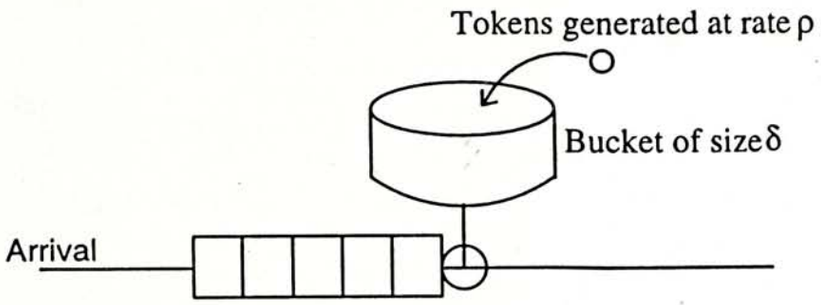

Figure 2.1: Structure of a leaky bucket.

The leaky bucket [15, 16, 17] is a access control mechanism for regulating the traffic entering a channel. Fig. 2.1 depicts the structure of a leaky bucket. There are two parameters associated with a leaky bucket : the token generation rate, ρ, and the bucket size, δ. Tokens generated at rate ρ are put into a bucket that can hold δ tokens. A packet must acquire an amount of tokens equal to its size from the bucket before it can be transmitted on the output. When the bucket fills up, any newly generated but unused token will be discarded. Usually there is a buffer associated to a leaky bucket. Packets arriving to an empty bucket will be buffered. The maximum transmission rate of a leaky bucket can either be bounded or unbounded. In the former, an additional rate control is imposed on the output traffic. In the following, we assume that the maximum transmission rate of a leaky bucket is unbounded. The practical meaning of this assumption is that the output rate can match the maximum aggregate input arrival rate

when the bucket is not empty.

2.2 (δ, ρ) Channel

Before going into the details of the characteristics of (δ, ρ) channels, let us review the characteristics of physical channels first. To describe the bandwidth of a physical channel, only one parameter is required. That is, the channel capacity, C. A physical channel is either in idle or busy state. Packets, one at a time, are transmitted at rate C in busy state and there is no transmission in idle state. A packet of size L takes L/C units of time to be transmitted. The transmission capacity during the idle state will be wasted. There is one kind of resources in a physical channel, which is the transmission capacity.

A (δ, ρ) channel is specified by two parameters δ and ρ, which limit the traffic that the channel can carry. We say a traffic flow with rate function $R(t)$ conforms to the (δ, ρ) specification, or according to Cruz's notation [18, 19], $R \sim (\delta, \rho)$ if

$$\int_x^y R(t)dt \leq \delta + \rho(y - x) \tag{2.1}$$

for all x, y satisfying $y \geq x$.

This specification is closely related to the concept the of leaky bucket. If $R(t)$ is the traffic rate function and $R \sim (\delta, \rho)$, then it can be immediately transmitted when regulated by a leaky bucket with parameters δ and ρ. Also, note that if $O(t)$ is the output traffic rate function of a leaky bucket with parameter δ and ρ, then $O \sim (\delta, \rho)$. Thus, if the access of a logical channel is controlled by a leaky bucket, then it is a (δ, ρ) channel. We expect that (δ, ρ) channels will be quite common, as it is very likely that the leaky bucket will be accepted as the access control method for ATM networks. For a (δ, ρ) channel, we can always imagine

that there is a leaky bucket controlling the access to it, although in general other mechanisms can be used to ensure conformance of the carried traffic to the (δ, ρ) specification. For example, in chapter 3, we shall see that a soft multiplexer can also enforce its output traffic to conform to the (δ, ρ) specification.

The characteristics of (δ, ρ) channels are listed in the following:

(1)Two resources : In a physical channel, there is only one kind of resources, the transmission capacity, and the resource will be wasted when there is no traffic entering the channel. However, in a (δ, ρ) channel, there are two kinds of resources : (1) the transmission capacity described by ρ, and (2) the power of saving up transmission capacity described by δ. When there is no traffic entering the channel, the transmission capacity at that time can be saved up for later use and at most capacity for transmitting δ data units can be saved up.

(2) Variable bit rate : If enough transmission capacity has been saved up, packets can be transmitted immediately, or in other words, the maximum transmission rate is infinite at that time. With infinite transmission rate, it is possible to transmit multiple packets simultaneously. However, when all stored transmission capacity has been used up, the maximum transmission rate is ρ. In contrast, only one packet can be transmitted by a physical link at a certain time. Moreover, If the channel capacity is C, a packet of size L may take L/C units of time to get transmitted. Therefore, packets will never be transmitted simultaneously.

(3)Bursty transmission : Basically, δ and ρ control the burstiness and the mean rate of the traffic entering the channel respectively. Because of the saving capability, the bit rate of the traffic entering the channel can fluctuate back and forth around the mean rate ρ. The larger the δ, the bigger the fluctuation allowed.

Let us compare the two systems shown in Fig. 2.2. In the upper system,

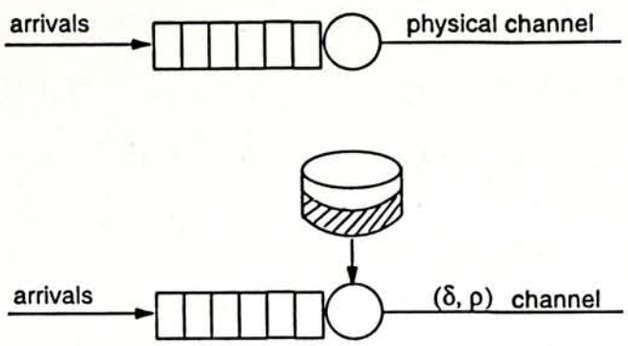

Figure 2.2: Comparison of physical and (δ, ρ) channel

a single traffic stream is transmitted by a physical channel with transmission capacity 1 pkt/s. In the lower system, the same traffic stream is transmitted by a (δ, ρ) channel with parameters (2, 1).

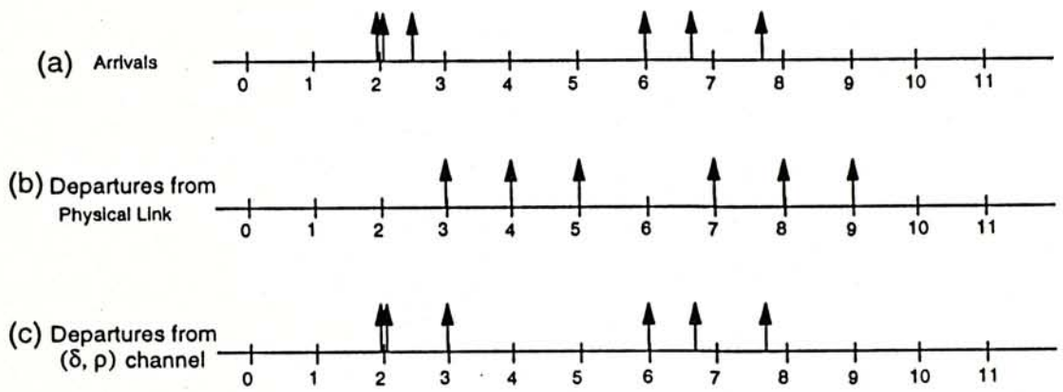

Figure 2.3: Comparison of physical and (δ, ρ) channel

With respect to the arrival pattern depicted in Fig. 2.3a, Fig. 2.3b shows the transmission time of the packets in upper system. Because we assume fixed-size packets, each packet suffers a fixed transmission delay, 1 unit time, in addition to the queueing delay. In the time period (0, 2), the physical channel is idle and the transmission capacity is being wasted. After time 2, it becomes busy and packets are transmitted at rate 1 pkt/s. Again, the channel becomes idle in the time interval (5, 6). We can see that a physical channel may alternate between

busy and idle states and produce a constant rate output when it is in busy state.

On the other hand, the transmission pattern on a (δ, ρ) channel can be quite irregular. From Fig. 2.3c, we can see that the packets arriving at time 2 can be transmitted immediately because there is enough stored transmission capacity. However, as stored transmission capacity has been used up, packet arriving at time 2.5 must be delayed until time 3. Otherwise, the (δ, ρ) specification will be violated. Basically, traffic will not be smoothed out if its burstiness does not exceed certain level.

2.3 Comparison of Deterministic VP's and (δ, ρ) VP's

In this section, we compare the use of deterministic VP's and (δ, ρ) VP's in ATM networks. Our comparison mainly focuses on two aspects:(1) the worst case performance of a VP and (2) the end-to-end statistical performance of the network. We found that while both deterministic VP's and (δ, ρ) VP's can provide worst case guarantee, (δ, ρ) VP's can yield better end-to-end statistical performance in ATM networks.

Deterministic VP's :

A deterministic VP, which is constant-bit-rate logical channel, can be described by a single parameter : the transmission capacity. For each physical link a deterministic VP traversed, a transmission capacity is reserved exclusively for this VP and there is no bandwidth sharing among the VP's on the same physical link. The summation of the transmission capacities of the VP's on a physical link cannot exceed the capacity of the physical link.

One advantage of deterministic VP's is that the performance (delay/backlog) of each VP is approximately fixed regardless of the congestion level of the network.

An drawback of deterministic VP's is that statistical multiplexing cannot be achieved when VP's are multiplexed on to physical links. As bursty traffic will be smoothed out before it enters a deterministic VP, it has no opportunity to try to use any excess bandwidth not used by other VP's on the same physical link.

(δ, ρ) *VP's :*

When (δ, ρ) VP's are used, besides the transmission capacity, each VP is also given a saving power. Although the physical bandwidth in the transmission-level network is fixed, a logical VP within a physical link can have varying bandwidth. Note that a deterministic VP can be considered as a (δ, ρ) VP with $\delta = 0$.

The traffic entering a (δ, ρ) channel must conform to a (δ, ρ) specification. When summation of the transmission capacities of the VP's on a physical link is less than the capacity of the physical link, it is possible to guarantee the performance of each VP if suitable hard multiplexing scheme is used to multiplex VP's on to physical link. One possible scheme to achieve this goal is packet-by-packet fair queueing (PFQ) which is the implementation of fluid-flow fair queueing (FFQ) in the real-packet scenario. Parekh and Gallager have proved that when the sources conform to the (δ, ρ) specification, the queueing delay and backlog over multiple PFQ/FFQ nodes are bounded.

Because (δ, ρ) VP's allow a certain level of bursty transmission which facilitates statistical multiplexing when several VP's are being multiplexed on to a physical link, (δ, ρ) VP's may yield better end-to-end statistical performance.

This better statistical performance is not achieved without cost. In the following, we discuss the drawbacks of using (δ, ρ) VP's. Traffic on deterministic VP's has been smoothed out at the access point and is not bursty. Therefore, only small buffers are required at the VP switches. However, for (δ, ρ) VP's, bursty traffic is given a chance to enter the VP subject to the (δ, ρ) specification. If the traffic of several VP's on a common physical link peaks together and there is no excess bandwidth, this bursty traffic must be buffered. Therefore, more buffers are needed in the VP switches when (δ, ρ) channels are used. Moreover, the larger the δ, the larger the buffers required.

When deterministic VP's are used, the delay on each VP is approximately fixed. However, when (δ, ρ) VP's are used, the VP-level network becomes more dynamic. And the end-to-end delay of a VP may change from time to time depending on the traffic situation. Assuming that FFQ hard multiplexing is used to multiplex VP on to physical link, Parekh [4, 5] has proved that the worst case delay/backlog on a VP is proportional to the δ of it.

The above discussion also relates to whether statistical guarantee or hard guarantee should be used in call admission control(CAC). Hard guarantee ensures that the end-to-end performance will never be worse than a certain value. On the other hand, statistical guarantee only ensures that the end-to-end performance will not be worse than a certain value with a high probability. If hard guarantee is used, it appears that dedicated network bandwidth needs to be allocated regardless of the statistical multiplexing gain in the network. Therefore, to maintain the benefit of multiplexing gain, statistical guarantee should be used in CAC. Under such a condition, VBR VP which can potentially provide better statistical performance may be the better choice.

2.4 A Simulation Study : The Effect of δ

In this section, we study the use of (δ, ρ) VP's and deterministic VP's by simulations. In the simulation model, we assume that there are 3 VP's being multiplexed on to a physical link. The statistical performance of the system (including the delay and backlog distributions) are examined when different VP's are being used.

Figure 2.4a and 2.4b show two systems, a and b, using deterministic VP's and (δ, ρ) VP's, respectively. We equate the physical resources in both systems by setting the transmission capacity of the physical links to 10000 cell/s. In system a a fixed transmission capacity, 3333.3 cell/s is allocated to each of the three VP's. On the other hand, besides this transmission capacity, one more parameter, δ, is associated to the VP's in system b. We vary δ to show the effect of it on the statistical performance. Note that when δ is set to zero, the two systems are equivalent.

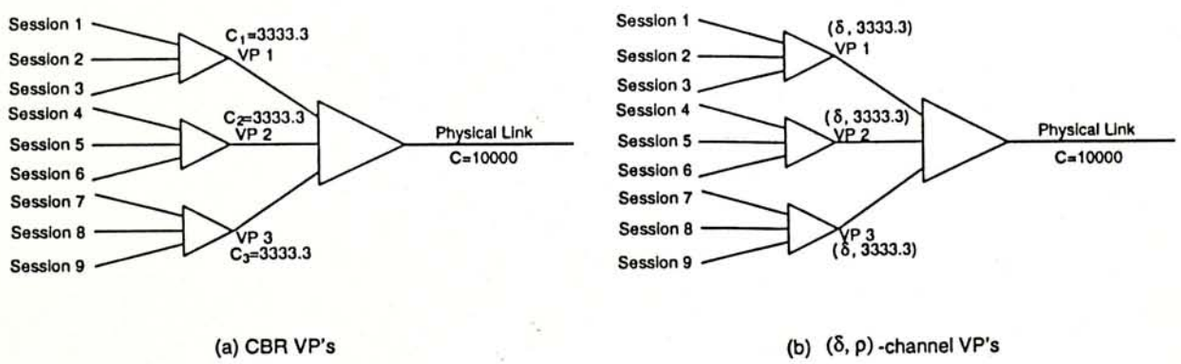

Figure 2.4: Simulation settings

We assume that the nine homogeneous sources are modeled by a Markov-modulated deterministic process (MMDP) which can be represented by the two

state continuous-time Markov chain shown in Fig. 2.5. The resulting traffic alternates between on and off periods which are exponentially distributed. Fixed-size Packets arrive with a fixed rate during an on period and no packet arrives during an off period. Therefore, each on-off traffic stream can be characterized by three parameters: mean duration of on periods, λ, mean duration of off-periods, μ, and peak rate (i.e. the rate when the source is on), P. Then the average source rate, r, can be obtained by :

$$r = \frac{\lambda P}{\lambda + \mu} \tag{2.2}$$

For each of the nine sessions, we set $\lambda = \mu = 0.03$ seconds. During an on-period, cells would arrive with rate $P = 2000\ cells/second$. The average rate can be calculated as in (2.2):

$$\begin{aligned} r &= \frac{0.03 \times 2000}{0.03 + 0.03} \\ &= 1000\ cells/s \end{aligned}$$

Therefore, the overall loading is 0.9.

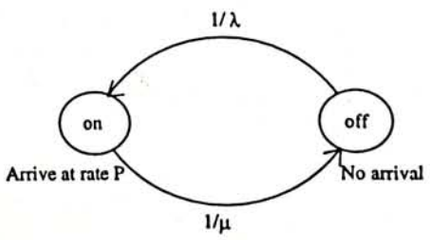

Figure 2.5: 2-state Markov chain for MMDP.

We assume that all multiplexers are FCFS multiplexers. For system a, all multiplexers are hard multiplexers, i.e. multiplexers with a CBR output channel. For system b, besides the hard multiplexer used to multiplex the three VP's on to the physical link, there are three soft multiplexers: multiplexers with a (δ, ρ)

channel as the output channel. A FCFS soft multiplexer does not distinguish between packets from different sessions. The oldest packets will be transmitted when there is enough stored transmission capacity for it.

When we increase δ, an obvious result is that the delay/backlog in the first-level multiplexing will decrease and the delay/backlog in the second-level multiplexing will increase. Therefore, it is inappropriate to compare the performance in a single node. In the following we focus on two performance metrics. The first one is the system delay, i.e. the time difference between when a cell enters and when it departs the system. Secondly, we examine the dynamic of the system backlog, i.e. the sum of the backlog in all multiplexers.

Figure 2.6 and 2.7 show the effect of δ on the distributions of the system backlog and the system delay. Note that the curves with $\delta = 0$ correspond to system a (deterministic VP's). We see that when δ increases, the whole distribution shifts left. When $\delta = \infty$, there will be no delay/backlog in the soft multiplexers and cells will arrive at the hard multiplexer directly. This corresponds to the case of single-level multiplexing and the best statistical performance is achieved. This result confirms that (δ, ρ) VP's can yield better statistical performance for ATM networks, when compared with deterministic VP's. Besides, it also shows that the two-level multiplexing of ATM networks may lower the statistical performance.

The effect of δ on the average system backlog and average system delay is shown in Fig. 2.8 and 2.9. When δ increases, the average backlog and delay will decrease and the effect of δ is diminishing. When δ tends to infinity, the average system backlog and delay will approach a constant, which corresponds to the case of single-level multiplexing.

In this simulation, we have only considered a single hard multiplexer. We find that the use of (δ, ρ) VP's facilitates statistical multiplexing in this hard multiplexer and thus better statistical performance is achieved. Our argument can be easily extended to the end-to-end case by considering that each of the hard multiplexer traversed by a VC can yield better statistical performance when each VP is a (δ, ρ) channel.

2.5 Summary of This Chapter

1. The traffic transmitted by a (δ, ρ) channel must conform to a (δ, ρ) specification. That is

$$\int_x^y R(t)dt \leq \delta + \rho(y - x)$$

for all x, y satisfying $y \geq x$, where R(t) is the traffic rate function.

2. There are two kinds of resources in a (δ, ρ) channel: (1) The transmission capacity and (2) the power of saving up transmission capacity.

3. Because (δ, ρ) VP's allow a certain level of bursty transmission, statistical multiplexing is achievable in lower level multiplexing. Compared with deterministic VP's, (δ, ρ) VP's can yield better end-to-end statistical performance of ATM networks.

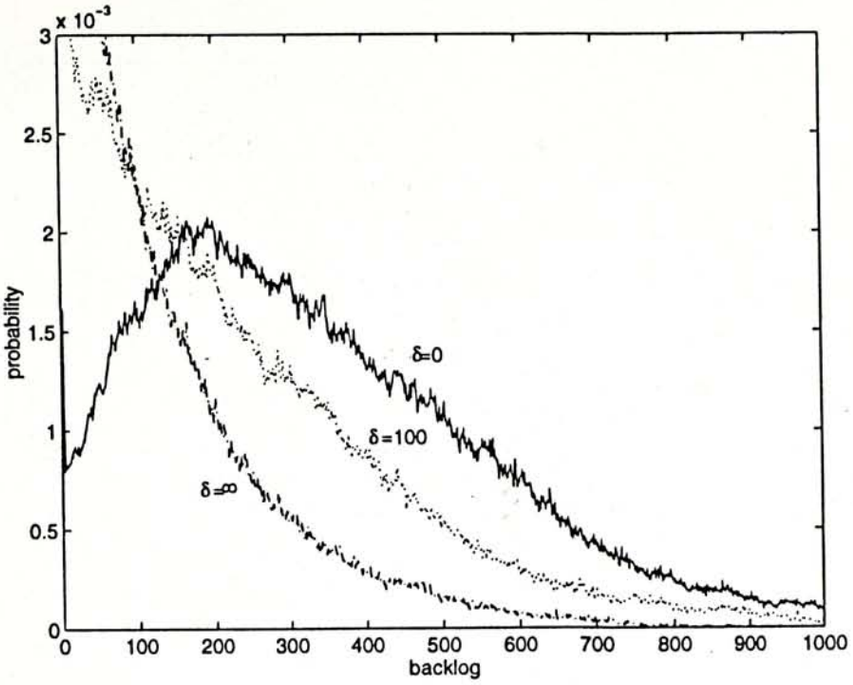

Figure 2.6: Effect of δ on the backlog distribution.

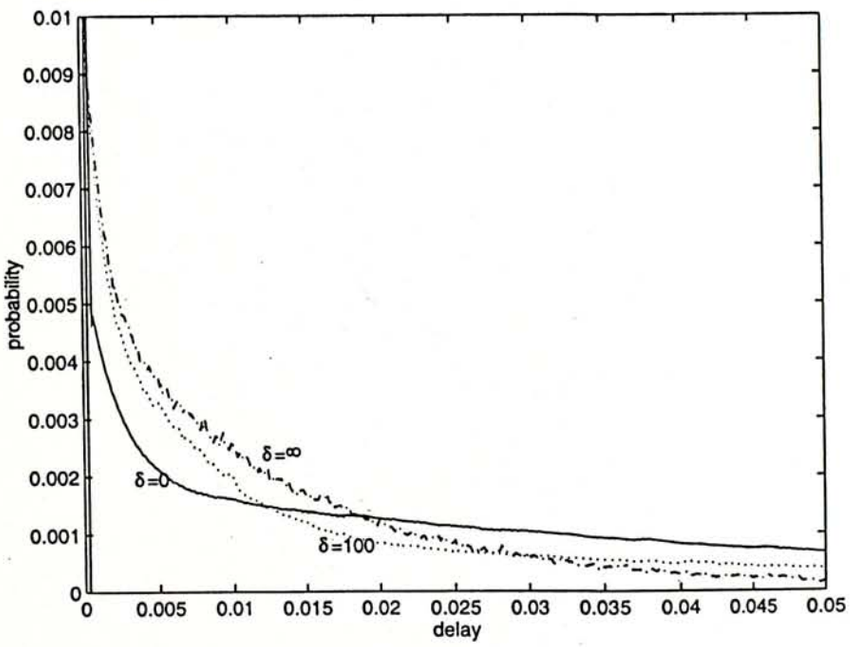

Figure 2.7: Effect of δ on the delay distribution.

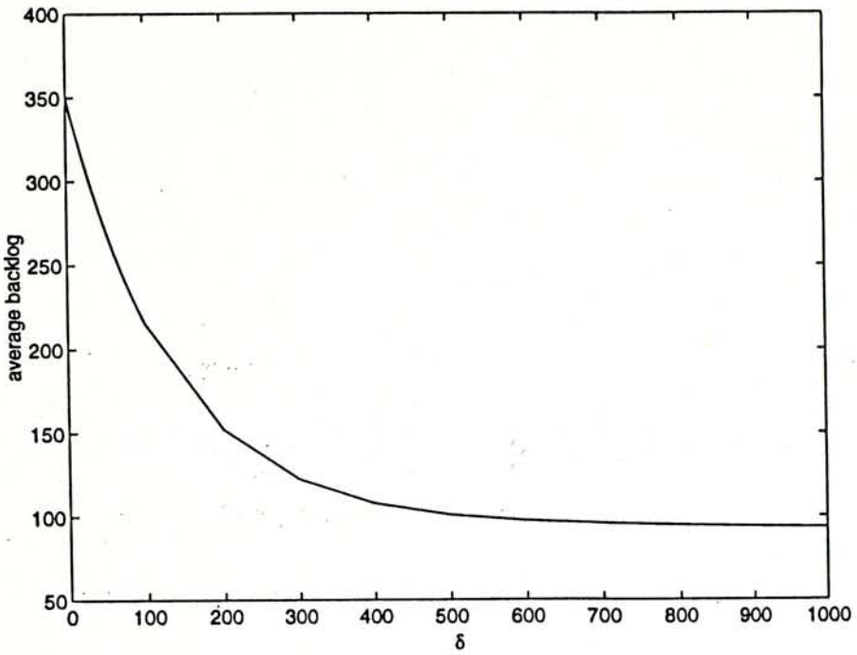

Figure 2.8: Effect of δ on the average backlog.

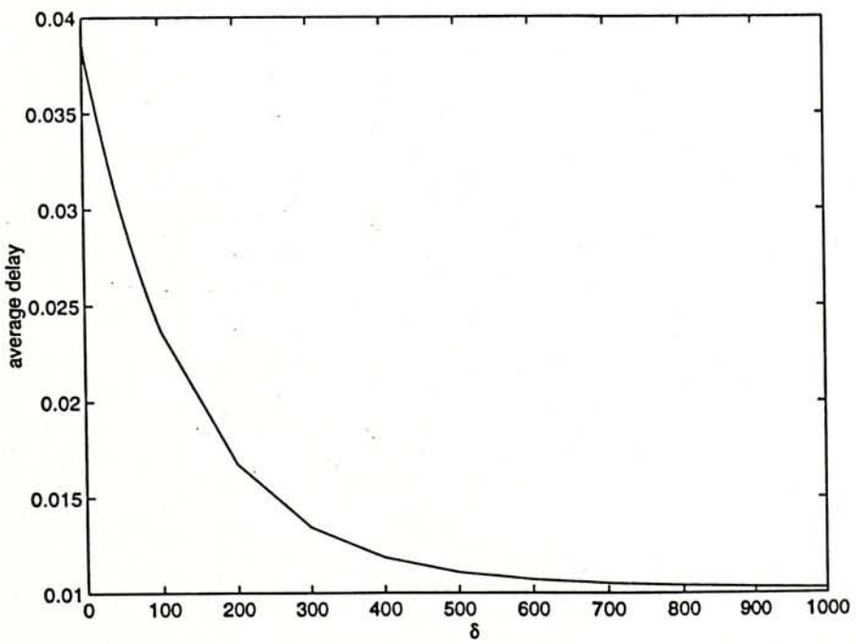

Figure 2.9: Effect of δ on the average delay.

Chapter 3

Soft-Multiplexing Scheduling Schemes

The better statistical performance of ATM network yielded by (δ, ρ) VP's motivates the study of soft-multiplexing scheduling schemes. The major responsibility of a soft multiplexer is to distribute the transmission resources to the sessions according to their characteristics and requirements. In the previous chapter, we have only considered the statistical performance of ATM network in the network level point-of-view. In this chapter, we look into the performance of individual sessions under different scheduling schemes in a soft multiplexer. We consider three scheduling schemes : first come first serve, fixed-resource allocation and excess token passing. We argue that there is a trade-off between performance guarantee to individual sessions and the average performance of all sessions being multiplexed. This statement is verified by simulations. We suggest the use of excess token passing schemes because they provide strong resource guarantee (hence performance guarantee), high output channel utilization and low

loss/delay in soft multiplexers.

This chapter is organized as follow. Section 3.1 outlines the issues and defines the goals of soft multiplexing. Section 3.2, 3.3 and 3.4 introduce and compare the three schemes: first come first serve, fixed-resource allocation and token passing, respectively. The performance of these schemes are compared using simulations in Section 3.5.

3.1 Issues in Soft Multiplexing

Multiplexing is a mechanism for sharing common resources. In hard multiplexing, the output channel is a physical channel which has only one kind of resources, the transmission capacity. Therefore, the issue in hard multiplexing is how to schedule the transmission of packets arriving from different sessions using this single resource. In soft multiplexing, the output channel is (δ, ρ) channel with two kinds of resources : (1) transmission capacity and (2) power of saving up transmission capacity. So, the issue in soft multiplexing is how to schedule the transmission of traffic arriving from different sessions using these two resources.

To clarify the picture, we can imagine that there is a leaky bucket associated with the soft multiplexer as shown in Fig. 3.1, indicating whether the soft multiplexer may produce output or not, if the (δ, ρ) specification of the output channel is to be conformed to. We refer to it as the *reference leaky bucket.* This reference leaky bucket has bucket size δ and tokens are generated at rate ρ. Note that picturing an reference leaky bucket is for conceptual understanding

of the constraint on the soft multiplexing only, and it may not exist in real implementation, as shall be seen later. An outgoing data unit from any input of the multiplexer will acquire tokens from the reference leaky bucket before it is transmitted on the output channel. As long as the reference leaky bucket is not empty, the soft multiplexer can generate output at infinite rate and the specification of the output channel will not be violated. On the other hand, when the reference leaky bucket is empty, the soft multiplexer should not produce output at rate higher that ρ. Otherwise the specification of the output channel will be violated. Sessions may contend for the tokens inside the reference leaky bucket. The soft multiplexer is responsible for distributing the tokens in the reference leaky bucket to the sessions such that their requirements can be satisfied.

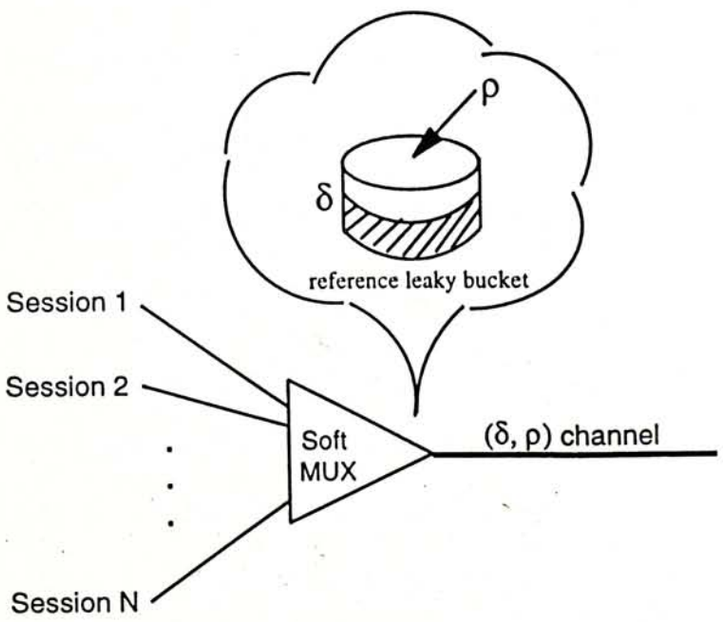

Figure 3.1: The reference leaky bucket.

Let us consider an interesting analogy in real life situation. We can imagine that the soft multiplexer is a man who has a constant-rate income, ρ, and a

pocket which can hold up to δ unit of money. One limitation is that he has to put his money inside his pocket and not anywhere else. This man has different kinds of expenses which have different characteristics and requirements. Some of them can be delayed a little bit, while others cannot. Some of the expenses are bursty(i.e. they are incurred once in a while, and each time there is a large sum of money involved), while others are constant expenses(i.e. they are incurred regularly). The problem is how to make use of the money in his pocket to fulfill the requirements of different expenses.

When designing a soft-multiplexing scheme, the following should be the objectives:

1. High utilization of the output channel: In soft multiplexing, bandwidth will be wasted only when the reference leaky bucket overflows. To ensure high utilization, unnecessary bucket overflow should be avoided.

2. Low loss/delay in the soft multiplexer: There are several metrics related to loss/delay performance : Average loss/delay, variance of loss/delay and worst case loss/delay.

3. Guaranteed performance to each session: An important requirement for multiplexers in integrated-service environment is that they must be able to guarantee the QOS of the sessions. To achieve this, resource must be reserved for individual sessions according to their characteristics and QOS requirements. Thus, if a scheme can provide resource guarantee, it can also provide performance guarantee to individual sessions.

Basically, the loss/delay performance in a multiplexer indicate the level of statistical multiplexing achieved. We find that, to provide a certain performance

guarantee, dedicated transmission resources must be reserved for individual sessions and this may affect the level of statistical multiplexing. Thus, there is a trade-off between performance guarantee to individual sessions (goal 3) and the overall average performance of the sessions being multiplexed (goal 2). Thus, how to maintain a balance between them is an essential issue when designing a soft-multiplexing scheduling scheme. In the following three sections, we introduce three different soft-multiplexing schemes to show this trade-off.

3.2 First Come First Serve (FCFS)

This is the scheme used in the simulation of Chapter 2. In the FCFS scheme, arrivals from any session may join a single queue. Packets in this queue are served in first in first out fashion, regardless of the sessions they belong to. The oldest packets in the head of queue will be transmitted when there are enough tokens in the reference leaky bucket.

Figure 3.2 shows a scenario with two sessions to illustrate how FCFS soft multiplexing works. Figure 3.2a shows the arrival process of the two sessions. Define $A(t)$ to be the cumulative arrivals to the system from time zero up to time t and $K(t)$ to be the cumulative number of tokens accepted into the reference leaky bucket from time zero up to time t. Figure 3.2b plots $A(t)$ and $K(t)$ when the arrival pattern in Fig. 3.2a is fed into a FCFS soft multiplexer with $\delta = 2$ and $\rho = 1$. Note that under packet-by-packet arrival traffic, $A(t)$ is a staircase function. We also assume that the tokens are generated in a fluid-flow manner(i.e. tokens are infinitely divisible and units of tokens are generated in a continuous fashion. When $K(t) = A(t)$, the reference leaky bucket is full. On

the other hand, when $A(t)$ goes beyond $K(t) + \delta$, the reference leaky bucket is empty and the traffic suffers a delay. In the time interval $(0, 1)$, one token has been discarded (i.e. certain transmission capacity of the output channel has been wasted because there is no backlog and no arrival during this time). At time 1, both the arrivals at time 1 can be served immediately using the tokens in the bucket. After time 1, tokens are generated at the rate of 1 token per unit time. Tokens are consumed immediately to clear up the backlog on a FCFS basis. The resulting departure process is shown in Fig. 3.2c.

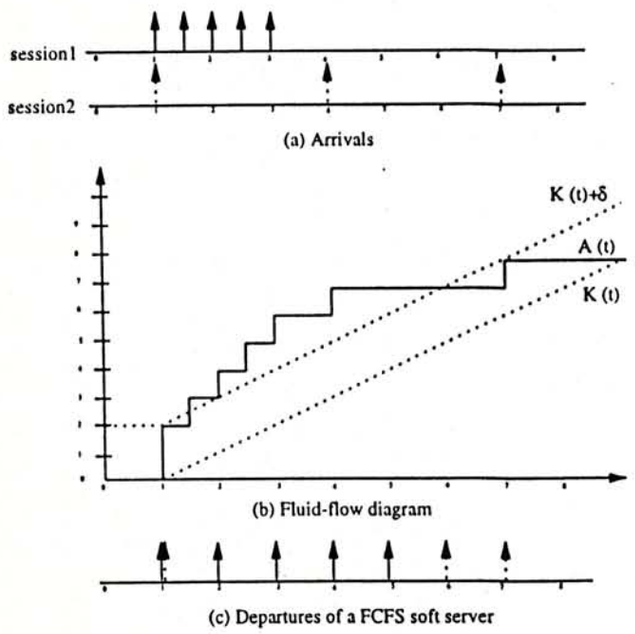

Figure 3.2: Characteristic of FCFS soft server.

In general, a soft multiplexer has the flexibility to decide whether to save or consume the tokens in the reference leaky bucket, even though there are packets waiting in the system. However, in the FCFS system, tokens will be consumed immediately whenever there are packets waiting. Thus, co-existence of tokens in the reference leaky bucket and packet backlog is impossible. Soft-multiplexing

schemes with this characteristic are said to be *greedy*[1]. More formally,

DEFINITION : *A greedy soft multiplexer will output a backlogged packet immediately if doing so does not violate the* (δ, ρ) *specification.*

A greedy soft server can achieve high utilization of the output channel. Moreover, because packet backlog is cleared as fast as possible, the average packet delay / loss of all sessions in the soft multiplexer is minimized. Note that given the same input traffic patterns, all greedy systems have identical average queue length and waiting times, regardless of the order in which the packets depart. Another advantage of FCFS is that it is easy to implement.

However, the drawback of FCFS is that it cannot provide any resource guarantee. A session generating large amount of traffic may use up all the tokens in the reference leaky bucket and cause delay/loss to other sessions. Thus, the QOS of individual sessions is not guaranteed at all. In the above example, let us suppose that session 1 carries computer data that is delay-tolerant and session 2 carries real-time data that cannot tolerate excessive delay. Even though these traffic characteristics are available to the soft server, by the nature of FCFS, it has no mean to provide any priority to the session 2's packets. As we can see, the long burst of session 1 has caused a serious delay to the session 2's traffic.

3.3 Fixed-resource Allocation

In FCFS, tokens in the reference leaky bucket are shared by all sessions and any arriving packet can consume it immediately. Therefore, no resource guarantee or performance guarantee is provided to individual sessions. In contrast, fixed-resource allocation can provide exclusive resource guarantee to sessions by giving

each session in the system a dedicated amount of saving power and transmission capacity. Every session can only make use of its own portion of resources. Denote the share of saving power and transmission capacity given to session i as δ_i and ρ_i. To make use of all the available resources, $\sum_{i=1}^{N} \delta_i = \delta$ and $\sum_{i=1}^{N} \rho_i = \rho$.

Conceptually, we can imagine that each session i may reference to a leaky bucket, LB_i, which has bucket size δ_i and token generation rate ρ_i. If there is token in LB_i, backlog of session i will be output immediately at the soft-multiplexer output. We say a session is *busy* if LB_i is not full and a session is *idle* if LB_i is full. If session i is idle, tokens passed to LB_i will be wasted.

Figure. 3.3 illustrates the mechanism of fixed-resource allocation. The same set of arrivals as in the previous example of FCFS is now fed into a fixed-resource allocation soft multiplexer with $\delta = 2$ and $\rho = 1$. Define $A_i(t)$ to be the cumulative arrivals of session i in the time interval $(0, t)$ and $K_i(t)$ the cumulative number of tokens accepted by LB_i in the time interval $(0, t)$. Assume that $\rho_1 = \rho_2 = 1/2$ and $\delta_1 = \delta_2 = 1$. The two graphs in Fig. 3.3b plot $A_1(t)$ and $K_1(t)$, $A_2(t)$ and $K_2(t)$ respectively. We can see that if session i is busy (i.e. LB_i not full) in a time interval, $K_i(t)$ will be a straight line with slope ρ_i in this time interval. In the time intervals (3, 4) and (6, 7), because session 2 becomes idle, tokens generated to LB_2 are wasted (slope of $K_2(t)$ is zero). The resulting departure process is shown in Fig 3.3c.

The merit of fixed-resource allocation is that resources can be allocated for each session according to their traffic characteristics and requirements. And thus, each session has guaranteed QOS independent to the traffic of other sessions. Consider the above example. In time 3, although there is a unused token in LB_2 and session 1 is backlogged, session 1 cannot grab this token as it is

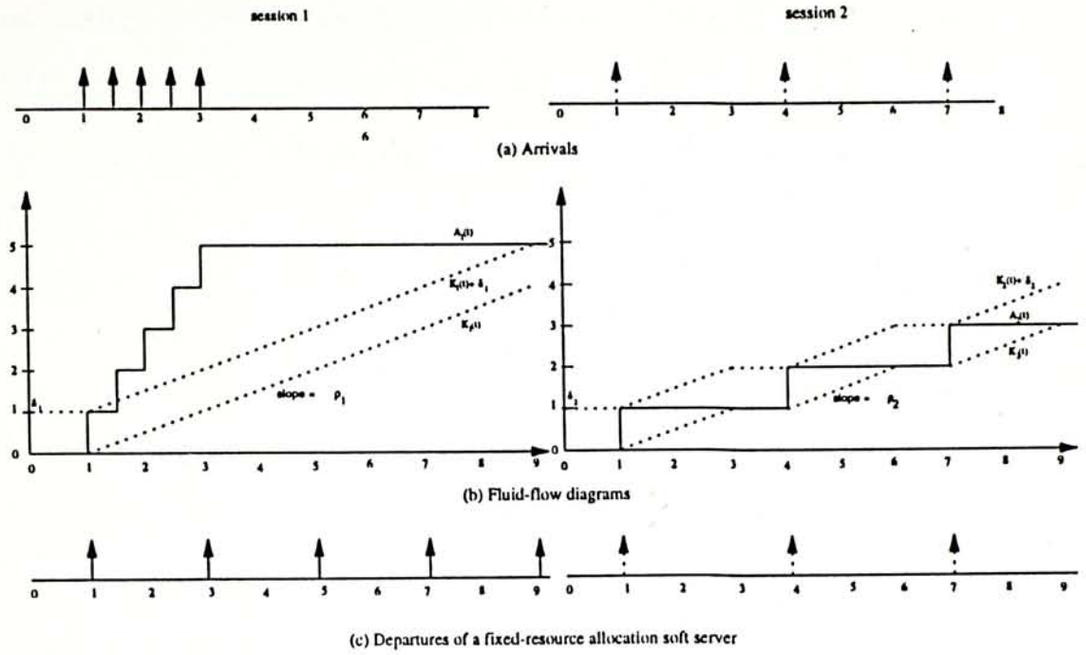

Figure 3.3: Fluid flow diagram for soft MUX with fixed-resources allocation.

reserved for session 2. Assume that session 1 and session 2 in the above example carry delay-insensitive and real time traffic respectively. Then session 2 can be served without any delay. Another advantage of fixed-resource allocation is that the implementation is straight forward.

However, an obvious drawback of fixed-resource allocation is poor utilization of output-channel bandwidth which causes unnecessary delay. A session may be wasting tokens while another active session may suffer undue delay and loss. Consider the above example again, in the time intervals (3, 4) and (6, 7), tokens generated to LB_2 are wasted. Actually, session 1 which is backlogged in these time intervals needs those excess tokens.

Let us examine the relationship between the LB_i's and the reference leaky bucket mentioned in Section 3.1 when fixed-resource allocation is used. Define $l_i(t)$ to be the number of tokens in LB_i at time t and $l(t)$ to be the number of

tokens in the reference leaky bucket at time t. Then $\sum_{i=1}^{N} l_i(t) < l(t)$ when any of the LB_i's overflows and some others are not full. This means that $l(t) - \sum_{i=1}^{N} l_i(t)$ tokens in the reference leaky bucket can never be used, or in other words, certain bandwidth of the output channel is wasted. Thus, discarding tokens in LB_i's implies wasting bandwidth of the output channel.

3.4 Excess Token Passing

This scheme is similar to fixed-resource allocation in the sense that each session also has guaranteed saving power δ_i and transmission capacity ρ_i. One major difference between them is that when session i becomes idle, other busy sessions may use the transmission capacity of session i, ρ_i, to avoid waste of resources. Or in other words, the token generation rate of a busy session can be passed to others.

Define $B(t)$ to be the set of busy sessions at time t and define $\rho_i(t)$ to be the effective token receiving rate of LB_i at time t. Then, if $i \notin B(t)$, $\rho_i(t) = 0$. If $i \in B(t)$, $\rho_i(t) = \rho_i + E$, where E is the instantaneous rate at which additional tokens are passed to LB_i. Based on this framework, we can define many different schemes depending on how the excess tokens from the idle sessions are distributed:

(1) Poorest session token passing: We say session i is the poorest session if $\frac{l_i(t)}{\delta_i}$ is the smallest in all i, where $l_i(t)$ is the number of token in LB_i at time t. In poorest session token passing, all excess tokens are always passed to the poorest session, which potentially will suffer much loss or delay than other sessions. If

session i is the poorest session at time t, then,

$$\rho_i(t) = \rho_i + \sum_{j \notin B(t)} \rho_j$$

(2) Uniform token passing: Excess tokens are distributed to the busy sessions uniformly. If $i \in B(t)$,

$$\rho_i(t) = \rho_i + \frac{\sum_{j \notin B(t)} \rho_j}{\text{number of busy sessions}}$$

(3) Rate proportional token passing: The larger the guaranteed transmission capacity a busy session has, the more excess tokens it gets. If $i \in B(t)$,

$$\rho_i(t) = \rho_i + \frac{\rho_i}{\sum_{j \in B(t)} \rho_j} \sum_{k \notin B(t)} \rho_k = \frac{\rho_i}{\sum_{j \in B(t)} \rho_j} \rho \tag{3.1}$$

According to (3.1), the tokens are passed to the busy sessions in a rate proportional way. Thus, we call this scheme the rate proportional token passing (RPTP).

To illustrate the mechanism of excess token passing, consider the example in Fig. 3.4. Because there are only two sessions, excess token from an idle session can only be passed to the other session. Therefore, all the above excess token passing schemes are identical when N=2. When both of the sessions are busy, tokens are passed to them at the guaranteed token generation rate, ρ_1 and ρ_2. However, in the time intervals (3, 4) and (6, 7) when session 2 is idle, tokens are passed to LB_1 at rate $\rho_1 + \rho_2 = \rho$.

A token-passing soft multiplexer can also provide strong resource guarantee to the sessions, just like the fixed-resource allocation can. And each session has a guaranteed performance independent to the traffic of other sessions. Adopting this scheme will not make any of the sessions worse off than adopting fixed-resource allocation. In addition, a session i's backlog will always be cleared at

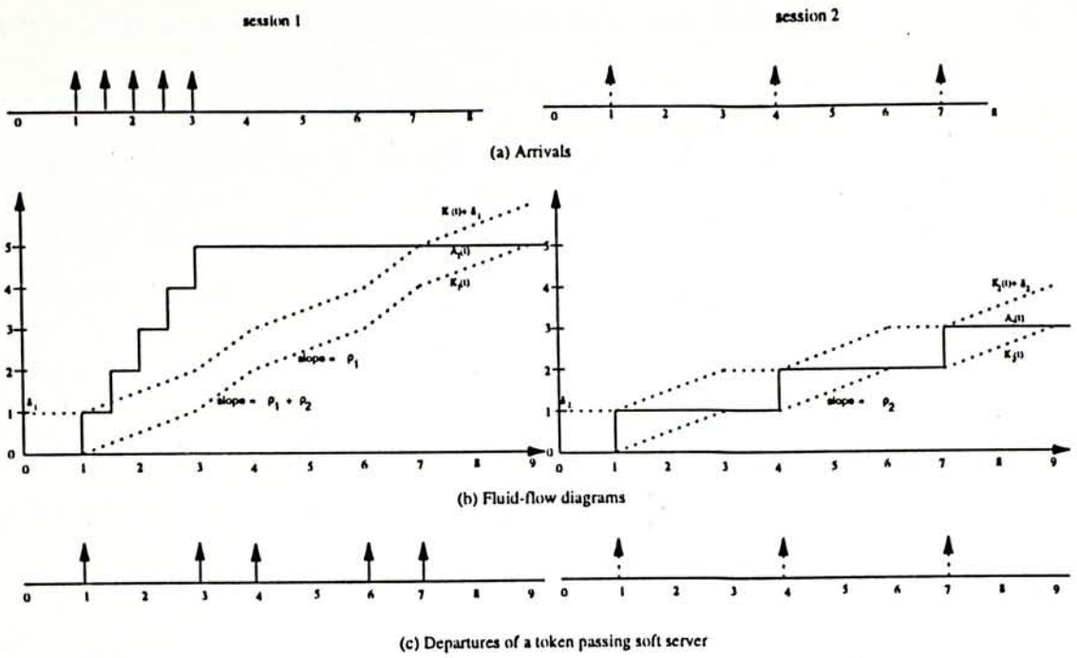

Figure 3.4: Characteristic of token passing soft server.

rate $\geq \rho_i$. In the above example, while session 2 can be served without any delay, delay of session 1 is lowered compared to the case of fixed-resource allocation.

Because no tokens of LB_i's are discarded if there is any busy sessions, $\sum_{i=1}^{N} l_i(t) = l(t)$, for all t. Thus, the reference leaky bucket overflows (i.e. the resources of the output channel are wasted) only when all sessions are idle, but this is unavoidable. Moreover, if there is packet backlog, no resources of the output channel will be wasted. Soft-multiplexing schemes with this characteristic are said to be *work conserving* [1], the definition of which is formally given below :

DEFINITION : *A soft multiplexer is said to be work conserving if when the reference leaky bucket is full, packet backlog, if any, must be cleared at a rate higher than or equal to* ρ.

A work conserving soft server ensures that no resources are wasted unnecessarily. Thus, excess token passing achieves high utilization of the output channel which ensures certain low level of loss/delay in the soft multiplexer.

Because, in a greedy soft multiplexer, coexistence of packet backlog and tokens in the reference leaky bucket is impossible, a greedy soft multiplexer (e.g. FCFS) is also work conserving. Fixed-resource allocation is neither greedy or work conserving, because it is possible that a session is backlogged while the reference leaky bucket overflows.

3.5 Simulation Results

From the discussion in the previous three sections, we have the following comments about the soft-multiplexing schemes:

(1) FCFS can achieve high output channel utilization and the best loss/delay performance but resource guarantee cannot be provided to the sessions.

(2) Fixed-resource allocation provides exclusive resource guarantee to the sessions. However, it suffers from low output channel utilization and poor loss/delay performance.

(3) Excess token passing schemes can provide resource guarantee to sessions, because resources dedicated to a session are re-distributed to other sessions only when they cannot be used by the session. In addition, they can also achieve high output channel utilization and good (not the best) loss/delay performance.

Let us generalize the above discussion and draw some conclusions. A work conserving scheme (e.g. excess token passing) can achieves high output channel utilization. Thus, we assert that a good soft multiplexer should be work conserving as it is unreasonable to waste output channel bandwidth when there is packet backlog.

The loss/delay performance indicates the level of statistical multiplexing achieved. As we have mentioned, there is a trade-off between the performance guarantee (achieved by resource guarantee) to individual sessions and the average performance of all sessions. Actually, we can define many different work conserving schemes with different combination of resource guarantee and the level of statistical multiplexing. In one extreme, greedy schemes (e.g. FCFS) provides the highest level of statistical multiplexing but no resource guarantee to individual sessions. In the other extreme, excess token passing schemes provides exclusive resource guarantee to individual sessions but only excess bandwidth of the idle sessions is available to others. Specifically, we can define a work conserving scheduling scheme with $\sum_{i=1}^{N} \delta_i < \delta$ and $\sum_{i=1}^{N} \rho_i < \rho$. The rest of the resources is shared by all sessions.

To verify the above qualitative statements, we have simulated the performance of FCFS, fixed-resource allocation and PSTP. In the first set of simulations, we focus on the overall average loss/delay performance of all sessions being multiplexed. In the second set of simulations, we examine the fairness and the capability of providing resource guarantee(or performance guarantee) by introducing a misbehaving session.

In the first simulation experiment, we assume that there are three homogeneous sessions modeled by Markov-modulated deterministic process (MMDP)

as shown in Fig. 2.5 of Chapter 2. For each of the three sessions, we set $\lambda = \mu = 0.016667$ seconds, which means that, on average, there would be 30 bursts in a second. During an on-period, cells would arrive with rate $P = 24000$ *cells/second*. The average rate can be calculated as in (2.2):

$$\begin{aligned} r &= \frac{0.016667 \times 24000}{0.016667 + 0.016667} \\ &= 12000\ cells/s = 5\ Mb/s \end{aligned}$$

These three sources are fed into three different multiplexers: FCFS, PSTP and fixed-resource allocation. The output (δ, ρ) channels of these multiplexers are identical and have parameters $(1200, \rho)$. Define the system load as

$$L = \frac{3r}{\rho} \tag{3.2}$$

We vary ρ to change the system load. For the case of PSTP and fixed-resource allocation, because the sources are homogeneous, each session may get guaranteed resources $(400, \rho/3)$, one third of the total resources.

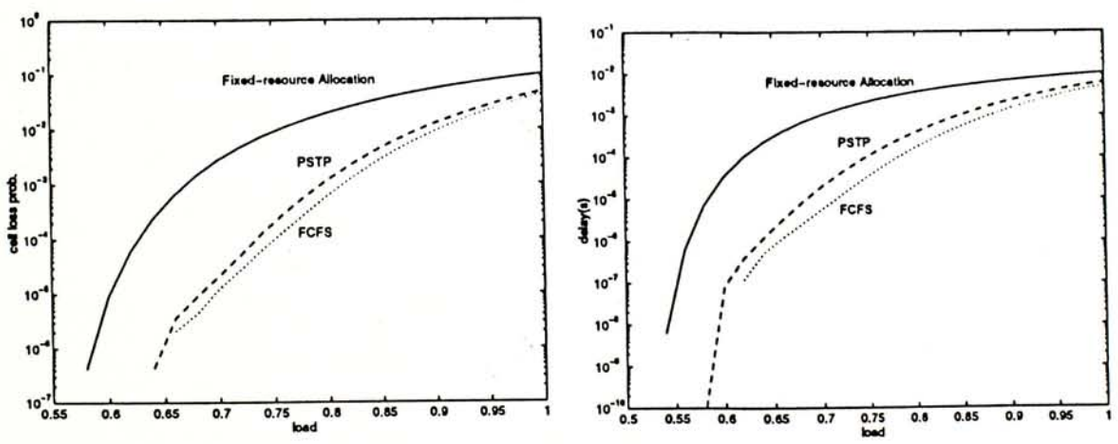

Figure 3.5: Comparison of soft multiplexing schemes.

Figure. 3.5 compares the cell-loss probability and average delay of the schemes under different system loads. We see that for the same system load $(\frac{3r}{\rho})$, both

cell-loss probability and average delay in PSTP and FCFS are better than those in fixed-resource allocation by one to two orders of magnitude. Therefore, for a fixed loss-probability or average delay requirement, the sustainable load of each session can be higher in PSTP and FCFS. Equivalently, for a fixed set of multiplexed sessions, an output channel with smaller capacity is needed. Let us compare FCFS and PSTP now. For both cell-loss probability and delay, FCFS has better performance than PSTP. This confirms our earlier statement that FCFS achieves the best loss/delay performance. However, the difference between the two curves is not significant. It seems that the strong resource guarantee (hence performance guarantee) that PSTP is capable of outweighs this little performance disadvantage.

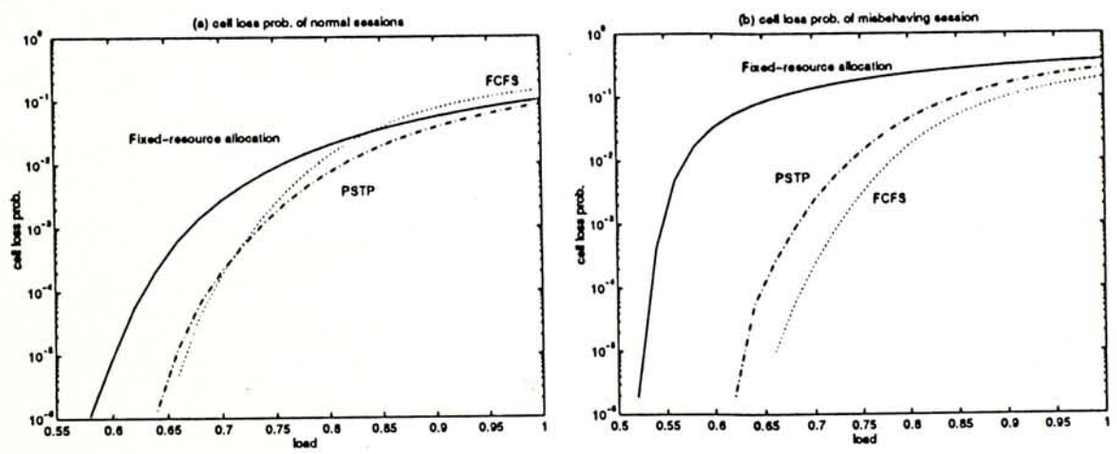

Figure 3.6: Effect of a misbehaving session.

We now examine the fairness of the schemes by considering a scenario in which one of the sessions misbehaves and injects long bursts into the output channel. We have set the mean on duration of the misbehaving session four times bigger than the normal sessions. Fig. 3.6a and 3.6b show the cell loss probabilities of the normal and the misbehaving sessions respectively. Let us

consider fixed-resource allocation first. Because of the firewall protection of fixed-resource allocation, the misbehaving session cannot grab the resources of the normal sessions. Therefore, while the misbehaving session suffers a high cell loss probability, the cell loss of the normal sessions is not affected. PSTP can also provide strong guarantee just like the fixed-resource allocation can. In addition, it allows the excess bandwidth of the idle sessions to be shared by others. Although most of the time the excess bandwidth is passed to the misbehaving session, the normal sessions may also get excess bandwidth occasionally. Therefore, when compared with fixed-resource allocation, both the normal sessions and the misbehaving session are better off. In the case of FCFS, as there is no protection for the normal sessions, the misbehaving session can grab most of the resources causing performance degradation of the normal sessions. Comparing the cell loss probabilities of the normal sessions to those of the misbehaving session, we find that they are approximately the same. Even though the average cell loss probability over all sessions is minimized, it is unfair that the normal sessions suffer such a big loss/delay. From the figures, we see that, under high load, the cell loss probabilities of the normal sessions in FCFS is worse than those of fixed-resource allocation.

3.6 Summary of This Chapter

1. The issue in soft multiplexing is how to schedule the transmission of the traffic arriving from different sessions using the two resources: (1) transmission capacity and (2) the power of saving up the transmission capacity.

2. FCFS can achieve the best output channel utilization and the best loss/delay performance. However, no resource guarantee can be provided to the sessions.

3. Fixed-resource allocation provides strong resource guarantee to the sessions. However, it suffers from low output channel utilization and bad loss/delay performance.

4. Excess token passing schemes can provide strong resource guarantee to the sessions. In addition, they can also achieve good (not the best) output channel utilization and loss/delay performance.

5. We show by simulation the trade-off between performance guarantee to individual sessions and the average performance of all sessions being multiplexed. We propose the use of excess token passing scheme because they provide exclusive resource guarantee while achieving a certain level of average performance.

Chapter 4

Analysis of Rate Proportional Token Passing

In the previous chapter, we have examined the trade-off between performance guarantee to individual sessions and the average performance of all sessions. And we find that excess token passing scheme can provide performance guarantee while achieving a certain level of overall average performance. However, whether a soft multiplexer can provide worst-case delay/backlog guarantee is an essential issue for it to be used in broadband networks. Up to now, we have not assumed any constraint on the source traffic. However, in order to bound the delay and backlog, some kind of traffic control must be imposed on the source traffic. Leaky bucket is a major candidate for such control. Parekh and Gallager [4] proved that when the sources are controlled by leaky buckets, the delay and backlog for an arbitrary session in a FFQ multiplexer is bounded. In [5], they further extended their results to bound the backlog and delay in multiple-multiplexer case. In this chapter, we show that similar worst-case guarantee can also be provided

by the RPTP soft multiplexer. To simplify the proof, we map the RPTP soft multiplexing to a fictitious hard multiplexing system to which Parekh's results are applicable. This allows us to derive the results of the RPTP system from Parekh's work.

In Section 4.1, we first construct this fictitious system. And then, in Section 4.2, we review the properties of the leaky-bucket-controlled sources. Section 4.3 provides a delay bound for all work conserving soft-multiplexing systems when all the sources are controlled by leaky buckets. Section 4.4 shows that the delay and backlog in an RPTP multiplexer are bounded and derives the scenarios under which the delay and backlog reach their maximums. To this end, we apply Parekh's results on the fictitious system constructed in the Section 4.1. Finally, Section 4.5 focuses on how to calculate the worst-case delay in an RPTP multiplexer based on the results established in Section 4.4.

4.1 The Fictitious System

In Chapter 3, we have introduced several soft multiplexing schemes. Some of these schemes, such as fixed-resource allocation and excess token passing schemes, give each session a minimum dedicated saving power which does not change over time. Let us call this kind of soft-multiplexing schemes *fixed-δ_i schemes*. In this section, we construct a fictitious hard-multiplexing system which is closely related to this kind of soft-multiplexing systems and facilitates our analysis of RPTP.

In a fixed-δ_i system, a session i cannot have a nonempty LB_i and backlog at the same time. Define $Q_i(t)$ to be the backlog of session i at time t and $l_i(t)$ to

be the amount of tokens in LB_i at time t. If $Q_i(t) > 0$, then $l_i(t) = 0$. On the other hand, if $l_i(t) > 0$, then $Q_i(t) = 0$. Therefore, instead of keeping track of both of these two variables separately, we may keep track of them together using only one variable. This is achieved by the fictitious system shown in Fig. 4.1. In the fictitious system, there are N fictitious queues, $FQ_1, \cdots, FQ_N$, one for each session. The service rate of FQ_i is always equal to the token receiving rate of LB_i, $\rho_i(t)$, An arrival from a session will enter both the fixed-δ_i system and the fictitious system and join to the corresponding fictitious queue. The states of the fictitious system and the fixed-δ_i system have a one-to-one correspondence relationship, as explained below.

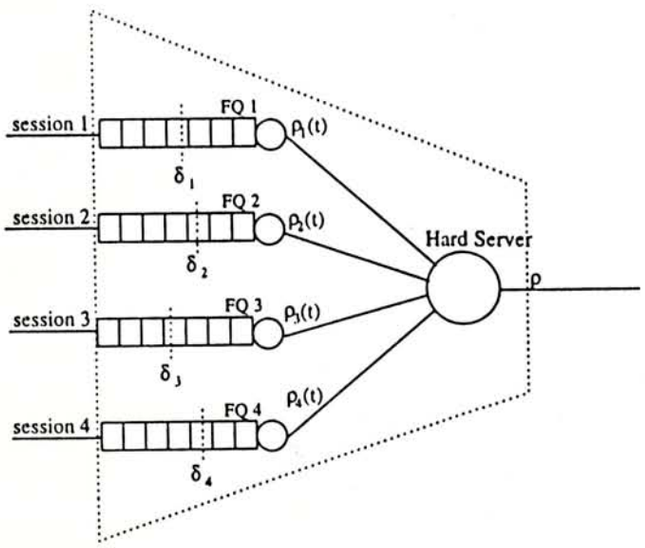

Figure 4.1: The fictitious hard-multiplexing system.

Let the occupancies of FQ_i at time t be $\bar{Q}_i(t)$. Then, $Q_i(t)$ and $\bar{Q}_i(t)$ are related by

$$Q_i(t) = max(\bar{Q}_i(t) - \delta_i, 0) \tag{4.1}$$

And the token level of LB_i, $l_i(t)$ is related to $\bar{Q}_i(t)$ by,

$$l_i(t) = max(\delta_i - \bar{Q}_i(t), 0) \tag{4.2}$$

When $\bar{Q}_i(t) = 0$, LB_i is full. When $\bar{Q}_i(t) > \delta_i$, tokens in LB_i has been used up and the backlog is $Q_i(t) = \bar{Q}_i(t) - \delta_i$. Define $\bar{S}_i(\tau, t)$ to be the accumulative output from FQ_i in $(\tau, t]$ and $K_i(\tau, t)$ to be the amount of tokens received by LB_i in $(\tau, t]$. Then,

$$\bar{S}_i(\tau, t) = K_i(\tau, t) \tag{4.3}$$

for all interval $(\tau, t]$. Because packets of the same session are served in a FIFO fashion in both systems, when a packet of session i enters the region from 0 to δ_i of FQ_i, it will depart from the fixed-δ_i system.

This fictitious system facilitates the analysis because we can keep track of $\bar{Q}_i(t)$ instead of both $Q_i(t)$ and $l_i(t)$. Besides this, the fictitious system appears like a hard-multiplexing system in which the issue is how to distribute the service capacity, ρ, to the sessions. This is similar to distributing the token generation rate, ρ, to LB_i's in fixed-δ_i systems. When a hard-multiplexing scheme is applied on the fictitious system, we then have a particular soft-multiplexing scheme in which the token receiving rate granted to LB_i at any time t is equal to the service rate of FQ_i at time t. Thus, given a scheduling discipline in a hard multiplexer, we can identify a corresponding fixed-δ_i soft-multiplexing scheduling discipline by doing the above mapping. As we shall see later, a lot of issues in fixed-δ_i system can be understood by borrowing results from previous work on the corresponding hard multiplexing system.

With respect to the definition of session business in the Section 3.4, if $\bar{Q}_i(t) > 0$ in the fictitious system, then session i is *busy*; otherwise, session i is *idle*. This

happens to also correspond to the usual definition of session business in hard-multiplexing systems.

If the hard-multiplexing scheme in the fictitious system is work conserving in the hard multiplexing sense, the associated soft-multiplexing scheme will also be work conserving in the soft multiplexing sense. Work conservation in the fictitious system with a hard-multiplexing scheme means that the backlog of the overall system will be cleared at rate ρ. This implies that, in the corresponding fixed-δ_i system, whenever there are non-full LB_i's, the aggregate token receiving rate of all LB_i's is equal to ρ, and thus, $\sum_{i=1}^{N} l_i(t) = l(t)$, for all t. If there is packet backlog, no resources of the output channel will be wasted. The resulting soft-multiplexing scheme is therefore work conserving.

Actually, the corresponding hard-multiplexing scheme of fixed-resource allocation is fixed-rate allocation in which dedicated transmission capacity is assigned to each session and there is no bandwidth sharing among sessions. One important observation is that a hard-multiplexing scheme and the corresponding soft one always have similar characteristics. For example, both fixed-rate allocation and fixed-resource allocation can provide strong guarantees to sessions but a common drawback of them is low output channel utilization.

The corresponding hard-multiplexing scheme of rate proportional token passing (RPTP) is fluid-flow fair queueing(FFQ). In FFQ, each session will be associated with a positive real number r_i which is the guaranteed service rate of session i. Note that $\sum_{i=1}^{N} r_i = C$, the channel capacity. A FFQ server distributes its service capacity in the following way:

$$\text{service rate of a session at time t} = \begin{cases} \frac{r_i}{\sum_{j \in B(t)} r_j} C, & \text{if busy} \\ 0, & \text{if idle} \end{cases} \tag{4.4}$$

where B(t) is the set of busy (i.e. backlogged) sessions at time t. According to the definition of RPTP in Section 3.5, we can see that the way a RPTP soft server passes the tokens is exactly equal to the way a FFQ hard server passes the service capacity. As FFQ is a work conserving hard-multiplexing scheme, RPTP is a work conserving soft-multiplexing scheme.

Another interesting example is the softened round-robin service discipline. In round robin, the server takes turn to serve a packet from the busy sessions. Therefore, the corresponding soft server may take turn to pass a token to those non-full LB_i's.

4.2 Leaky-Bucket-Controlled Sources

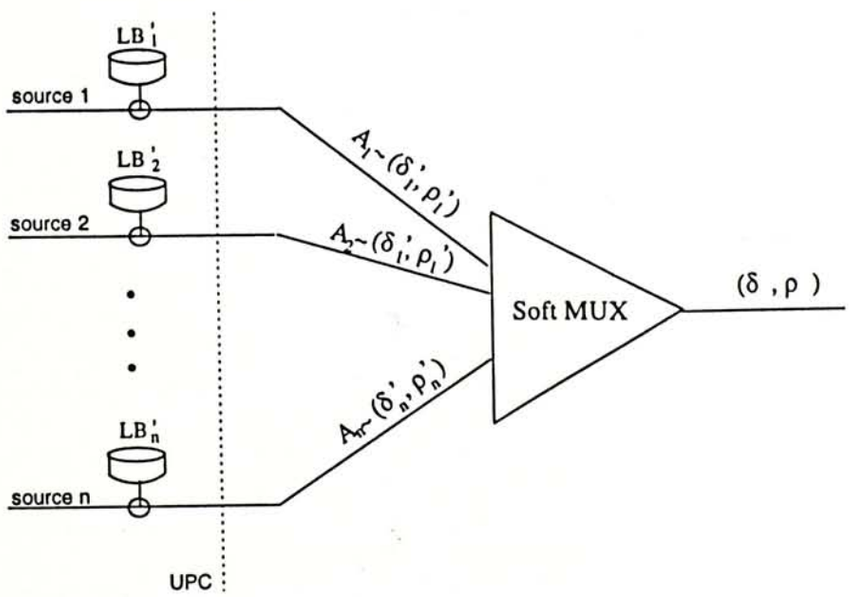

Figure 4.2: Soft multiplexing of N leaky-bucket controlled sources.

Fig. 4.2 shows that N leaky-bucket-controlled sources go through a single soft multiplexer in which they are multiplexed on to a (δ, ρ) channel. Denote the

leaky bucket controlling source i as LB'_i, and its bucket size and token generation rate as δ'_i and ρ'_i respectively. Assuming infinite incoming link capacity, we let $A_i(\tau, t)$ be the amount of session i flow that *leaves* LB'_i and arrives at the soft MUX in time interval $(\tau, t]$. Then,

$$A_i(\tau, t) \leq \delta'_i + \rho'_i(t - \tau) \tag{4.5}$$

We say a session i is greedy if it uses as many tokens as possible at LB'_i. If session i begins to be greedy at time τ, then,

$$A_i(\tau, t) = l'_i(\tau) + (t - \tau)\rho'_i$$

Let $K'_i(\tau, t)$ be the total number of tokens accepted by LB'_i in the interval $(\tau, t]$(it does not include the excess tokens that are discarded when the bucket overflows and the tokens that session i starts out with at time τ). Also let $l'_i(t)$ be the number of tokens in LB'_i at time t. Because the amount of departed traffic from LB'_i is equal to the number of tokens used, we have

$$A_i(\tau, t) = l'_i(\tau) + K'_i(\tau, t) - l'_i(t) \tag{4.6}$$

Note that this equation is true for all leaky buckets even though the token generation rate is not a constant. Recall that, in RPTP, each session i has a leaky bucket, LB_i, with bucket size δ_i. And the effective token receiving of these LB_i's may change over time because excess tokens from idle sessions are available to others. Thus, similar equation as (4.6) also exists for LB_i. If $S_i(\tau, t)$ is the accumulative output of session i from the soft multiplexer in $(\tau, t]$ and $K_i(\tau, t)$ is the number of tokens accepted by LB_i in $(\tau, t]$, we can get similar equations:

$$S_i(\tau, t) = l_i(\tau) + K_i(\tau, t) - l_i(t) \tag{4.7}$$

where that $l_i(t)$ is the amount of tokens in LB_i at time t.

4.3 Delay Bound for All Work Conserving Soft Multiplexers

In this section, we prove that when the sources are leaky-bucket-controlled, the delay of an arbitrary session in a work conserving soft multiplexer is bounded. Define the *system backlogged period* to be the time interval in which there is backlog of any session in the system. The following theorem states that if $\sum \rho_i' < \rho$, the length of a system backlogged period is bounded in a work conserving soft multiplexer.

THEOREM 1: *For a work conserving soft multiplexer, if* $\sum \rho_i' < \rho$, *the length of a system backlogged period is bounded by*

$$\frac{\sum_{i=1}^{N} \delta_i' + \delta}{\rho - \sum_{i=1}^{N} \rho_i'}$$

PROOF: Assume the system backlogged period is from t_1 to t_2. Then, from the definition of work conservation (see Section 3.4), no token of the reference leaky bucket is discarded in $(t_1, t_2]$. Define $l(t)$ to be the amount of tokens in the reference leaky bucket at time t. Then,

$$\sum_{i=1}^{N} A_i(t_1, t_2) = l(t_1) + \rho(t_2 - t_1) - l(t_2)$$

Since, from (4.5), $A_i(t_1, t_2) \leq \delta_i' + \rho_i'(t_2 - t_1)$, we get,

$$\begin{aligned} \sum_{i=1}^{N} \delta_i' + \sum_{i=1}^{N} \rho_i'(t_2 - t_1) &\geq l(t_1) + \rho(t_2 - t_1) - l(t_2) \\ &\geq \rho(t_2 - t_1) - \delta \end{aligned}$$

By rearranging the above equation, we get,

$$t_2 - t_1 \leq \frac{\sum_{i=1}^{N} \delta_i' + \delta}{\rho - \sum_{i=1}^{N} \rho_i'} \tag{4.8}$$

Since the delay of a packet from any session must be less than the system backlogged period, this lemma bounds the delay in all work conserving soft multiplexers.

Note that when $\rho = \sum_{i=1}^{N} \rho_i'$, the bound is infinite. Also note that this bound becomes higher when δ increases. It is intuitively not true because larger saving power of the output channel should yield lower delay of the input packets. The reason for this is that a work conserving soft multiplexer has the flexibility of choosing to save tokens rather than serve backlog, when the reference leaky bucket is not full. In the worst case, at time t_1, the beginning of the system backlog period $l(t_1)$ =0. It may take time $T = \frac{\delta}{\rho}$ to save up a whole bucket of tokens without serving any packet. Then, at $t_1 + T$, because of the definition work conservation, backlog must be cleared at rate larger than or equal to ρ. Assume it takes another T' time to clear the backlog. Then

$$\sum_{i=1}^{N} \delta_i' + \sum_{i=1}^{N} \rho_i'(T + T') \geq \rho(T')$$

Rearranging the above equation, we can bound the system backlog period, $T+T'$, by

$$T + T' \leq \frac{\sum_{i=1}^{N} \delta_i' + \rho T}{\rho - \sum_{i=1}^{N} \rho_i'}$$

Therefore, the smaller the *delta*, the shorter the T, and thus the lower the bound of the system backlog period.

4.4 The All-Greedy Bound in a RPTP Multiplexer

Although Theorem 1 bounds the delay in all work conserving soft multiplexers, this is not a tight bound in general. In this section, we derive the scenarios under which the delay and backlog for an arbitrary session in a RPTP soft multiplexer reach their maximum, when the sources are controlled by leaky buckets. In the analysis, we make use of the mapping between RPTP and FFQ via the fictitious system, and borrow results from Parekh's work.

Denote the worst-case backlog and worst-case delay of session i as Q_i^* and D_i^*, respectively. The following theorem states exactly when Q_i^* and D_i^* occur.

THEOREM 2: *For every session i in a RPTP multiplexer, D_i^* and Q_i^* are achieved when every session is greedy starting at time zero, the beginning of a system busy period.*

This theorem is identical to the main result of Parehk and Gallager, except that the target system is RPTP instead of FFQ. The interesting point of this theorem is that all $Q_1^* \cdots Q_N^*$ and $D_1^* \cdots D_N^*$ will be achieved under the same arrival pattern.

PROOF :

Recall that the corresponding fictitious system of RPTP is actually an FFQ system. According to Parekh's result, the worst-case backlog $\bar{Q}_i^*$ and worst-case delay $\bar{D}_i^*$ of session i in the fictitious system are achieved when every session is greedy starting at time zero. Then, from (4.1), we get,

$$Q_i^* = \bar{Q}_i^* - \delta_i$$

and we have proved half of Theorem 2.

Based on this backlog bound, we can bound D_i^* by $\frac{Q_i^*}{\rho_i}$ because session i has guaranteed token receiving rate ρ_i and traffic of a session is served in FIFO fashion. However, this is not the bound as stated in the theorem. To prove the rest of the theorem, a more lengthy derivation is needed.

Before proceeding, we need to introduce some notation. Let $\hat{A} = \{\hat{A}_1 \cdots \hat{A}_N\}$ be the set of arrival functions in which all the sessions are greedy from time 0, the beginning of a system busy period. Moreover, we also label other functions under such arrival pattern with a hat. For example, $\hat{S}_i(\tau, t)$ is the accumulative output of session i from the soft multiplexer in the interval $(\tau, t]$ under $\hat{A}$.

First of all, we apply one important lemma in [4] (Lemma 10 in page 355) to the fictitious system of RPTP, which is actually a FFQ system. After changing some notations, we rewrite Lemma 10 in [4] as the following lemma. For the proof of this lemma, please refer to [4].

LEMMA 1: *Consider the fictitious system of RPTP. Suppose the time t is contained in a session i busy period that begins at time τ, then*

$$\hat{\bar{S}}_i(0, t-\tau) \leq \bar{S}_i(\tau, t)$$

regardless what the arrival pattern is.

Moreover, from (4.3), we can rewrite Lemma 1 as,

$$\hat{K}_i(0, t-\tau) \leq K_i(\tau, t) \tag{4.9}$$

Consider an arbitrary arrival $A = A_1, \cdots, A_N$. Suppose a session i busy period starts at time τ and at time t^*:

$$D_i(t^*) = \max_{t \geq \tau} D_i(t)$$

According to the definition of delay,

$$A_i(\tau, t^*) - S_i(\tau, t^* + D_i(t^*)) = 0$$

Denote, $d_i^* = t^* - \tau$. From lemma1,

$$\hat{K}_i(0, d_i^* + D_i(t^*)) \leq K_i(\tau, t^* + D_i(t^*)) \tag{4.10}$$

Also, from the definition of greedy session,

$$A_i(\tau, t^*) \leq \delta_i' + \rho_i'(t^* - \tau) = \hat{A}_i(0, t^* - \tau) \tag{4.11}$$

Combining (4.10) and (4.11), we get,

$$\hat{A}_i(0, d_i^*) - \hat{K}_i(0, d_i^* + D_i(t^*)) \geq A_i(\tau, t^*) - K_i(\tau, t^* + D_i(t^*))$$

From, (4.7),

$$\begin{aligned} &\hat{A}_i(0, d_i^*) - \hat{S}_i(0, d_i^* + D_i(t^*)) - \hat{l}_i(d_i^* + D_i(t^*)) + \hat{l}_i(0) \\ &\geq A_i(\tau, t^*) - S_i(\tau, t^* + D_i(t^*)) - l_i(t^* + D_i(t^*)) + l_i(\tau). \end{aligned}$$

Because both $\hat{l}_i(0)$ and $l_i(\tau)$ are equal to δ_i, they will be canceled out and we get

$$\begin{aligned} &\hat{A}_i(0, d_i^*) - \hat{S}_i(0, d_i^* + D_i(t^*)) - \hat{l}_i(d_i^* + D_i(t^*)) \\ &\geq A_i(\tau, t^*) - S_i(\tau, t^* + D_i(t^*)) - l_i(t^* + D_i(t^*)) \end{aligned}$$

Note that $l_i(t^* + D_i(t^*))$ must be 0. Otherwise $D_i(t^*)$ is not the maximum delay under $A = A_1, \cdots, A_N$. Therefore,

$$\hat{A}_i(0, d_i^*) - \hat{S}_i(0, d_i^* + D_i(t^*)) \geq 0$$

and

$$\hat{D}_i(d_i^*) \geq D_i(t^*)$$

Therefore, via the fictitious system, we can derive results of the RPTP system by borrowing Parekh's works on FFQ. Because the concept of the fictitious system is general, besides RPTP, it can also be used to explore other soft-multiplexing systems by studying the corresponding hard-multiplexing systems.

4.5 Calculation of the Worst-Case Delay in a RPTP Multiplexer

Up to now, we have proven that the delay and backlog in a single RPTP multiplexer for an arbitrary session are bounded, when the sources follow the leaky-bucket constraint. Moreover, we know exactly under what scenario the worst case will occur, that is when all sessions begin to be greedy at time 0. Theoretically, when given (δ_i, ρ_i) and (δ_i', ρ_i'), for all i, we can find D_i^* and Q_i^* deterministically by assuming the all-greedy arrival pattern. In this section, we focus on finding D_i^* when (δ_i', ρ_i'), and (δ_i, ρ_i) for all i are given. However, we found that the close form of D_i^* differs from case to case and the number of cases will grow with N linearly. For illustration, we first consider the multiplexing of 2 leaky-bucket controlled sessions onto a output channel with $\delta = 0$ (i.e. hard multiplexing).

Let L(i) be the ith session which ends its busy period under the assumption that all sessions start being greedy at time zero and define *ending order* to be the sequence of sessions $L(1), \cdots, L(N)$. Then, when N=2, the ending order

can be easily determined by comparing $\frac{\delta_i'}{\rho_i-\rho_i'}$ $(i=1,2)$ and choosing the session with smaller but positive $\frac{\delta_i'}{\rho_i-\rho_i'}$ as $L(1)$. Note that at least one of $\frac{\delta_i'}{\rho_i-\rho_i'}$ $(i=1,2)$ must be positive because $\rho_1'+\rho_2'<\rho$ and $\rho_1+\rho_2=\rho$. Assume s_1 and s_2 are in ending order. Then, in the period $(0,e_1)$, s_1 and s_2 are served at constant rate ρ_1 and ρ_2 respectively. For s_1 to be the first session ending its busy period, ρ_1 must be larger than ρ_1' and thus, $e_1=\frac{\delta_1'}{\rho_1-\rho_1'}$. After e_1, s_1 keeps arriving at rate ρ_1'. Because of the fluid flow assumption, s_1 will be served with arrival rate ρ_1' and s_2 will make use of the rest bandwidth $\rho-\rho_1'$ which must be larger than ρ_2'. And thus e_2 can be calculated by

$$e_2=\frac{\delta_2'+e_1(\rho-\rho_1'-\rho_2)}{\rho-\rho_1'-\rho_2'}$$

To find D_1^*, consider Fig. 4.3a and measure the maximum horizontal distance between $\hat{A}_1$ and $\hat{S}_1$. Recall that we indicate the functions under the all greedy arrival pattern with a hat. By simple geometry, we get,

$$D_1^*=\frac{\delta_1'}{\rho_1}. \tag{4.12}$$

Again, for D_2^*, we can measure the maximum horizontal distance between $\hat{A}_2$ and $\hat{S}_2$. However, the way to calculate D_2^* may be different depending on which of the three cases, as shown in Fig. 4.3b, s_2 is in. By inspecting each of these three cases, we get

$$D_2^*=\begin{cases}\frac{\delta_2'}{\rho_2}, & \text{if}\rho_2\geq\rho_2', e_1\geq\frac{\delta_2'}{\rho_2}\\ e_1-\frac{\rho_2 e_1-\delta_2'}{\rho_2'}, & \text{if}\rho_2<\rho_2', e_1\geq\frac{\delta_2'}{\rho_2}\\ \frac{\delta_1'+\delta_2'}{\rho-\rho_1'}, & \text{if}e_1<\frac{\delta_2'}{\rho_2}\end{cases} \tag{4.13}$$

Note that in case 3 (i.e. $\rho_2<\rho-\frac{\delta_1'\rho+\rho_1'\delta_2'}{\delta_1'+\delta_2'}$), D_2^* is independent of ρ_1 and ρ_2. This means that even though $\rho_2=0$, s_2 can still have worst-case delay bound

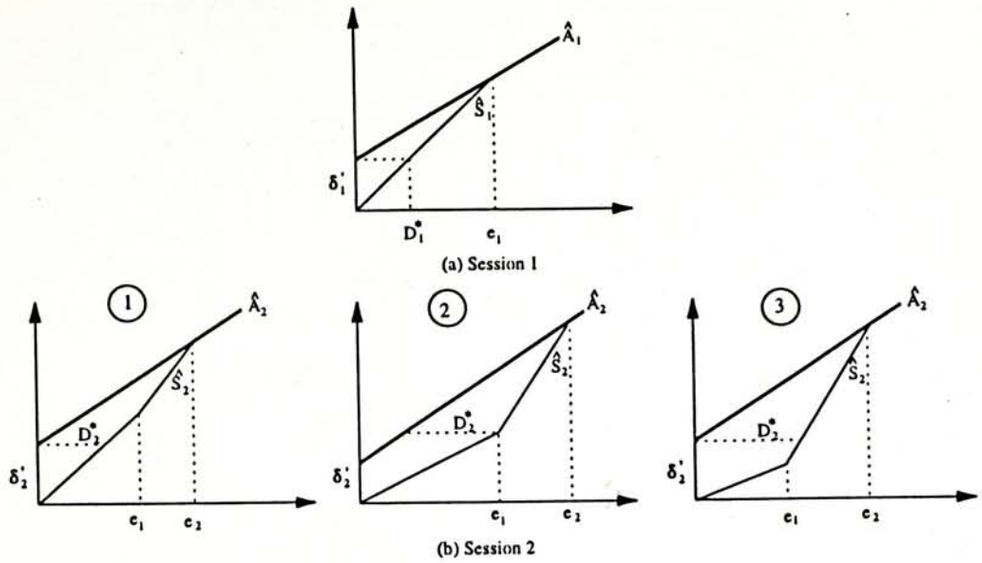

Figure 4.3: Worst-case delay when N=2.

$D_2^* = \frac{\delta_1'+\delta_2'}{\rho-\rho_1'}$. For $\rho_2 \geq \rho - \frac{\delta_1'\rho+\rho_1'\delta_2'}{\delta_1'+\delta_2'}$, s_2 is either in case 1 or 2 depending on the value of ρ_2. If $\rho_2 \in (\rho - \frac{\delta_1'\rho+\rho_1'\delta_2'}{\delta_1'+\delta_2'}, \rho_2')$, s_2 is in case 2 and if $\rho_2 \geq \rho_2'$, s_2 is in case 1. Therefore, when $(\delta_1', \rho_1'), (\delta_2', \rho_2')$ and ρ are given, using (4.12) and (4.13), we can plot the worst-case delay curves of s_1 and s_2 by varying ρ_1 and ρ_2. Note that $\rho_2 = \rho - \rho_1$. Fig. 4.4 shows a pair of worst-case delay curves, given $(\delta_1', \rho_1') = (50, 4.5)$, $(\delta_2', \rho_2') = (40, 3.5)$ and $\rho = 10$.

Equations (4.12) and (4.13) can also be used to find the feasible region of resource allocation when the delay requirements of each session are given. For example, in Fig. 4.5, if we want $D_1^* < a$ and $D_2^* < b$, the feasible region of ρ_1 is (c, d).

Before considering the case of $\delta > 0$ (i.e. soft multiplexing), let us see one interesting characteristic of fixed-δ_i systems. That is, the busy period of session i is independent to δ_i. It can be easily seen by the observation that the busy period of session i is equal to the backlogged period of FQ_i in the fictitious system,

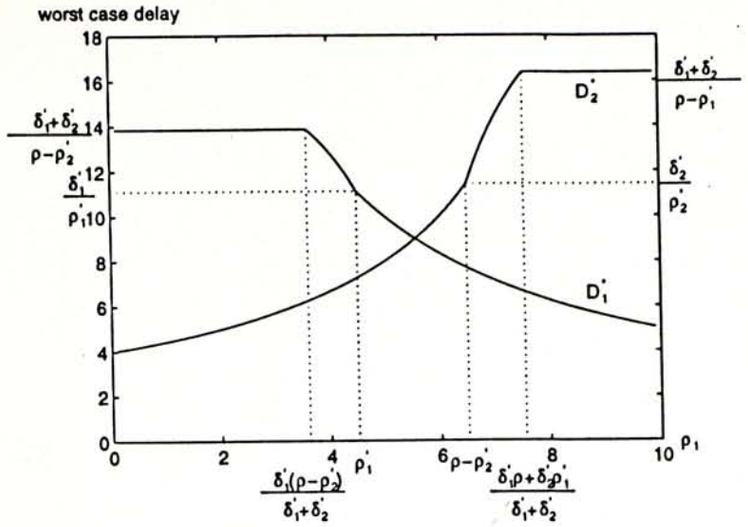

Figure 4.4: Worst-case delay VS ρ_1.

and the operation of the fictitious system is independent to δ_i's. Assume that , given (δ_i', ρ_i') and (δ_i, ρ_i) for all i, $1, 2, \cdots, N$ are arranged in the ending order and e_i is the time the ith session ends its busy period when all N sessions start greedy at time zero. Then, no matter how we change δ_i's, the ending order and e_i's will not change. We refer to this property as *order preservation.* Fig. 4.6 shows that when we change δ_i, the curve $\hat{K}_i + \delta_i$ will shift vertically and e_i is not affected.

Therefore, according to the order preservation property, we can still use the method we mentioned before to determine the ending order when N = 2. That is to choose the session with the smallest but positive $\frac{\delta_i'}{\rho_i - \rho_i'}$ as $L(1)$. Assume s_1 and s_2 are in ending order. Then, e_1 and e_2 are still $\frac{\delta_1'}{\rho_1 - \rho_1'}$ and $\frac{\delta_2' + e_1(\rho - \rho_1' - \rho_2)}{\rho - \rho_1' - \rho_2'}$ respectively. Because $\hat{S}_i(t)$ will be equal to $\hat{K}_i(t) + \delta_i(t)$ when $\hat{K}_i(t) + \delta_i(t) < \hat{A}_i(t)$, D_i^* can be found by measuring the maximum horizontal distance between $\hat{A}_i(t)$ and $\hat{K}_i(t) + \delta_i(t)$. Accordingly, we find,

$$D_1^* = max(\frac{\delta_1''}{\rho_1}, 0) \tag{4.14}$$

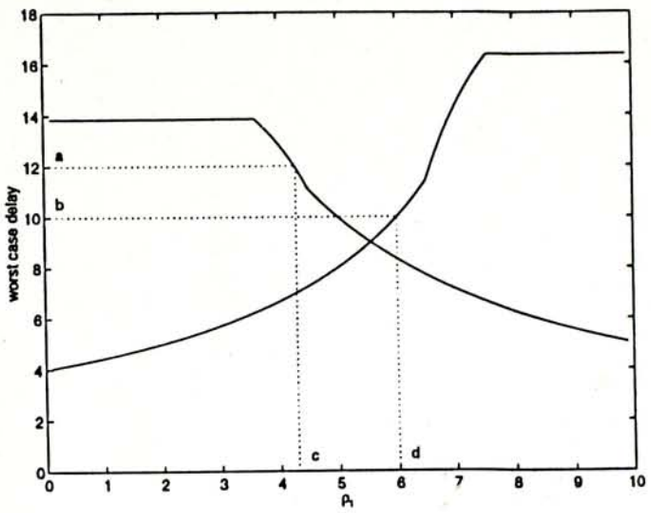

Figure 4.5: Feasible resources allocation region.

and

$$D_2^* = \begin{cases} max(\frac{\delta_2''}{\rho_2}, 0), & \text{if} \rho_2 \geq \rho_2', e_1 \geq \frac{\delta_2''}{\rho_2} \\ max(e_1 - \frac{\rho_2 e_1 - \delta_2''}{\rho_2'}, 0), & \text{if} \rho_2 < \rho_2', e_1 \geq \frac{\delta_2''}{\rho_2} \\ max(\frac{\delta_1'' + \delta_2''}{\rho - \rho_1'}, 0), & \text{if} e_1 < \frac{\delta_2''}{\rho_2} \end{cases} \tag{4.15}$$

where $\delta_i'' = \delta_i' - \delta_i$.

When $(\delta_1', \rho_1'), (\delta_2', \rho_2')$ and (δ, ρ) are given, using (4.14) and (4.15), we can plot the worst-case delay curves of s_1 and s_2 by varying ρ_1 and δ_1. Note that $\rho_2 = \rho - \rho_1$ and $\delta_2 = \delta - \delta_1$. Fig. 4.7 shows a pair of the worst-case delay curves, with $(\delta_1', \rho_1') = (50, 4.5)$, $(\delta_2', \rho_2') = (40, 3.5)$ and $(\delta, \rho) = (40, 10)$.

As we can see in Fig. 4.7, δ_i and ρ_i are substitution of each other. To achieve a certain delay requirement, various combinations of δ_i and ρ_i can be used. Thus, the feasible region of bandwidth allocation is a 2-dimensional plane, when delay requirements of all sessions are given.

We have considered the case of $N = 2$ and found that to determine the ending order, only one comparison is needed. After that, D_i^* can be found by close form equations and the close form equation of D_2^* contains 3 different

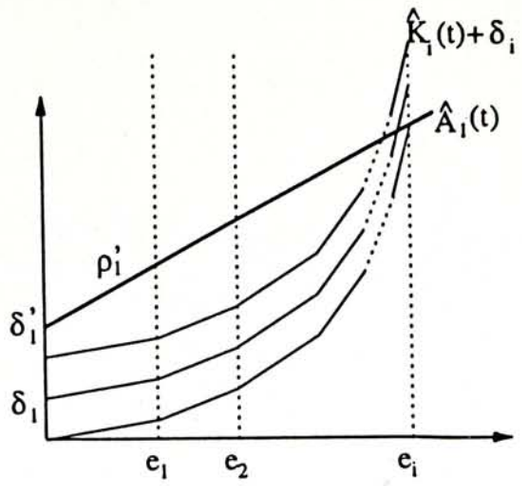

Figure 4.6: Order preservation property of soft multiplexing.

cases. In general, for $N > 2$, to sort out the ending order, $\frac{N(N-1)}{2}$ comparisons are needed and the close form equation of D_i^* ($1, \cdots, N$ are in ending order) may contain $2i - 1$ cases. Therefore, when N is larger, it is not easy to find D_i^* and further investigation is required.

4.6 Summary of This Chapter

1. A fictitious system, which provides a one-to-one mapping of hard-multiplexing schemes and a special type of soft multiplexing called the fixed-δ_i multiplexing schemes, has been constructed. Via this fictitious system, we found that a lot of issues in soft multiplexing can be understood by borrowing results from previous works on hard multiplexing. For example, the corresponding hard-multiplexing scheme of RPTP is actually FFQ. This allows us to derive results of the RPTP system from Parekh's work.

2. When the sources are leaky-bucket controlled, the delay of a packet from

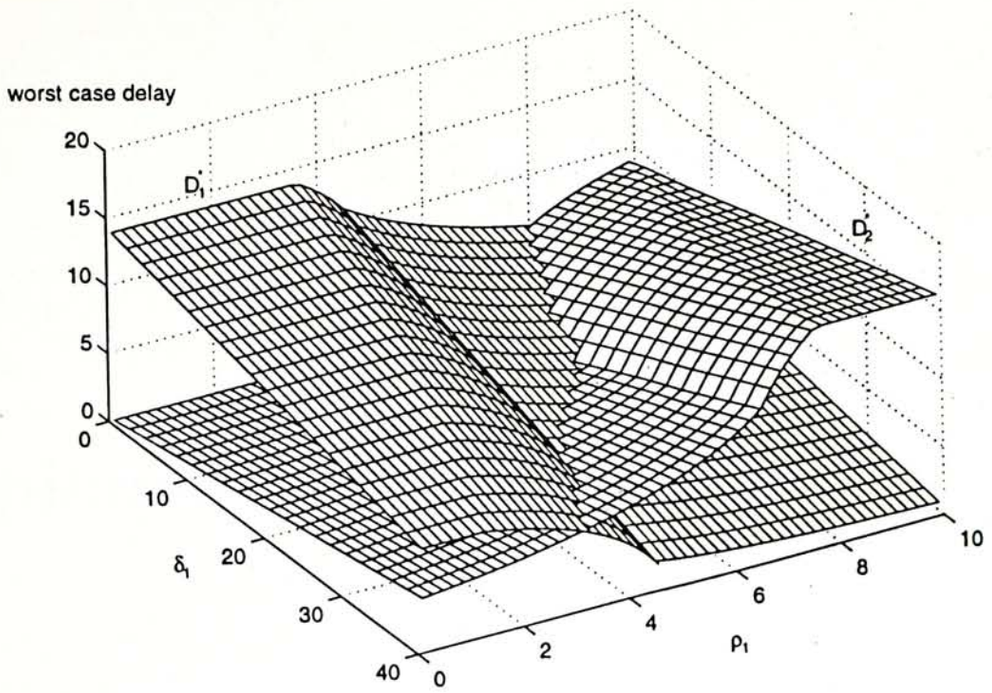

Figure 4.7: worst-case delay when varying δ_1 and ρ_1.

any session i in a work conserving soft multiplexer must be bounded by

$$\frac{\sum_{i=1}^{N} \delta_i' + \delta}{\rho - \sum_{i=1}^{N} \rho_i'}$$

3. For every session i in RPTP system, the worst-case delay, D_i^*, and worst-case backlog, Q_i^*, are achieved when every session is greedy starting at time zero, the beginning of a system busy period.

4. The calculation of the worst-case delay, D_i^*, has been considered. Close form equations have been obtained when N=2. For larger N, using numerical algorithm may a better way.

Chapter 5

Implementation of RPTP

The main difference between the implementations of soft and hard multiplexing is that the way they produce output. In hard multiplexing, outgoing packets, one at a time, are transmitted on the output physical channel. If the capacity of the output channel is C, a packet of size L may take $\frac{L}{C}$ units of time to be transmitted. So, packets will never depart simultaneously. In soft multiplexing, the output channel is a (δ, ρ) channel, which has infinite transmission rate when there is stored transmission capacity. So, it is possible that multiple packets will depart immediately.

To see why the transmission rate of a logical channel is not necessary bounded, consider Fig. 1.2 in Chapter 1 again. Note that the two levels of multiplexing will occur at the same output port of a VC switch. For a packet to depart the first multiplexer and reach the input of the second multiplexer, it may only involve internal data transferring (such as the memory address and the VPI of a packet). Thus, the processing required by the output from the first multiplexer is negligible. Since, from the VP-level point of view, a packet is transmitted on

the VP when it departs the first multiplexer, we can consider that there is no peak rate limit on the VP.

In this chapter, we will describe two different implementations of RPTP soft multiplexers. The first one is called virtual time implementation and the second one is called leaky-bucket implementation.

5.1 Virtual Time Implementation

Because FFQ assumes that the traffic is infinitely divisible, it is not applicable in actual packet-based traffic situations. Demers has defined a packet-by-packet fair queueing(PFQ) in which packets are served according to the sequence they depart the idealized fluid-flow system. Later, the virtual time implementation of PFQ is investigated by Parehk. The approach he used is to associate each arriving packet with a time stamp, the virtual finishing time (VFT), which indicates the virtual time on which this packet may depart the FFQ system. After that, packets from all sessions are inserted into a common output queue in which packets are sorted according to their VFT's. The server always picks the packet at the head of output queue (i.e. the packet with the smallest VFT) to serve. Although the VFT is calculated upon arrival, the real departure time will vary as the advance ("ticking") rate of the virtual time is not fixed. In this section, we may adopt a similar approach. That is, to calculate the VFT of a packet in the RPTP system upon arrival. In the calculation of VFT, we may make use of the method used in [4] to find the VFT of a packet in the corresponding fictitious system (FFQ system) first and then convert the result back to the VFT in the RPTP system.

Before discussing the virtual time implementation of RPTP, let us examine some other properties of RPTP system under the actual packet-based situation. If both sessions i and j are busy at time t, according to the definition of RPTP their normalized token receiving rates at time t are equal. That is,

$$\frac{\rho_i(t)}{\rho_i} = \frac{\rho_j(t)}{\rho_j} = \frac{\rho}{\sum_{k \in B(t)} \rho_k}, i, j \in B(t) \tag{5.1}$$

And if a session k is idle at time t, then its normalized token receiving rate at time t is zero. That is,

$$\frac{\rho_k(t)}{\rho_k} = 0, k \notin B(t) \tag{5.2}$$

Based on the property above, we define a virtual time function $V(t)$ with the following characteristics :

(1) $V(0) = 0$

(2) $\frac{dV(t)}{dt} = \frac{\rho_i(t)}{\rho_i} = \frac{\rho}{\sum_{i \in B(t)} \rho_i}, i \in B(t)$

This $V(t)$ has two meanings. First, it indicates the rate at which the tokens are passed to a busy session in the RPTP system. Second, it also indicates the rate at which the backlog of a fictitious queue is being served. From the definition of $V(t)$, we can see that $V(t)$ is a piecewise linear function which changes its slope whenever the set of busy sessions, B(t), changes. When all the sessions are busy, the slope of V(t) is 1. And the virtual time will advance faster if there are fewer busy sessions. Define $B(t_1, t_2)$ to be the set of sessions which are busy throughout the interval $(t_1, t_2]$. If session i is in $B(t_1, t_2)$, the normalized service it receives in the fictitious system in $(t_1, t_2]$, $\frac{\bar{S}_i(t_1,t_2)}{\rho_i}$, or the normalized number of tokens it receives in RPTP system in $(t_1, t_2]$, $\frac{K_i(t_1,t_2)}{\rho_i}$, is equal to the growth of the virtual time of the system. That is,

$$V(t_2) - V(t_1) = \int_{t_1}^{t_2} \frac{\rho_i(t)}{\rho_i} dt = \frac{\bar{S}_i(t_1, t_2)}{\rho_i} = \frac{K_i(t_1, t_2)}{\rho_i}, i \in B(t_1, t_2) \tag{5.3}$$

Let us consider two types of events in the fictitious system : arrival and departure events. Define t_i to be the time at which the ith event occurs. Consider an arbitrary interval (t_k, t_{k+1}). Since there is no state change between events (i.e. $B(t)$ is unchanged in (t_k, t_{k+1}) for any k), $V(t)$ is linear with slope $\frac{\rho}{\sum_{i \in B(t_1,t_2)} \rho_i}$ in this time interval. Thus, we arrive at the update equation of the V(t).

$$V(t_{k+1}) = V(t_k) + \frac{t_{k+1} - t_k}{\sum_{i \in B(t_k,t_{k+1})} \rho_i} \rho \tag{5.4}$$

Therefore, virtual time is updated only when an event in the fictitious system occurs.

Suppose that the kth packet of session i, p_i^k, arrives at time a_i^k and departs the fictitious system at time d_i^k. Define $\bar{F}_i^k$ to be the virtual finishing time of p_i^k in the fictitious system. Then $\bar{F}_i^k = V(d_i^k)$. Because session i must be busy in the period $(a_i^k, d_i^k]$, from (5.3), we get,

$$\bar{F}_i^k = V(d_i^k) = V(a_i^k) + \frac{\bar{Q}(a_i^k) + L_i^k}{\rho_i} \tag{5.5}$$

in which $\bar{Q}(a_i^k)$ is the backlog of FQ_i upon arrival of p_i^k and does not include the packet just arrived. Again from (5.3),

$$\bar{Q}(a_i^k) = \max\{\bar{F}_i^{k-1} - V(a_i^k), 0\}\rho_i \tag{5.6}$$

Combining (5.5) and (5.6), we arrive to the formula to calculate $\bar{F}_i^k$,

$$\bar{F}_i^k = \frac{L_i^k}{\rho_i} + max(V(a_i^k), \bar{F}_i^{k-1}) \tag{5.7}$$

Define F_i^k to be the virtual finishing time of p_i^k in the RPTP system. Recall that p_i^k will depart the soft-multiplexing system when it enters the region form 0 to δ_i of FQ_i in the fictitious system. Therefore F_i^k can be calculated by,

$$F_i^k = V(a_i^k) + \max\{\frac{\bar{Q}(a_i^k) + L_i^k - \delta_i}{\rho_i}, 0\} \tag{5.8}$$

When $\bar{Q}(a_i^k) + L_i^k - \delta_i \leq 0$, p_i^k has already entered the region from 0 to δ_i of FQ_i upon arrival and it can depart immediately. So, $F_i^k = V(a_i^k)$. When $\bar{Q}(a_i^k) + L_i^k - \delta_i > 0$, p_i^k needs to wait $\frac{\bar{Q}(a_i^k)+L_i^k-\delta_i}{\rho_i}$ amount of virtual time for it to enter the region from 0 to δ_i of FQ_i. Therefore, $F_i^k = V(a_i^k) + \frac{\bar{Q}(a_i^k)+L_i^k-\delta_i}{\rho_i}$.

Combining (5.5) and (5.8), we find that $\bar{F}_i^k$ and F_i^k are related by the following equation,

$$F_i^k = max(\bar{F}_i^k - \frac{\delta_i}{\rho_i}, V(a_i^k)) \tag{5.9}$$

Therefore, upon arrival of p_i^k, we can calculate F_i^k by (5.7) and (5.9) and insert p_i^k to the output queue according to F_i^k.

As we have mentioned, $V(t)$ needs to be updated when either an arrival or departure event in the fictitious system occurs. The problem now is how to keep track of the departure events as the queues are fictitious. Let $\bar{F}_{min}$ be the smallest VFT in the fictitious system and $Next_f(t)$ be the prediction time of the next departure event in the fictitious system at time t, assuming there is no arrival in $(t, Next_f(t))$. By assuming this, there will be no state change in $(t, Next_f(t))$. From (5.4), $Next_f(t)$ can be found by

$$\bar{F}_{min} - V(t) = \frac{Next_f(t) - t}{\sum_{i \in B(t)} \rho_i} \rho$$

We can keep track of the next departure event by setting up an interrupt at $Next_f(t)$. Of course, if it turns out that there is an arrival in $(t, Next_f(t))$, we may need to recalculate $Next_f(t)$ and change the interrupt time.

Now, packets are sorted in the output queue according to their VFT's calculated from (5.7) and (5.9). One major difference between the virtual time implementations of soft and hard multiplexing is the way the output queue produces output. In hard multiplexing, to avoid wasting service capacity, packets

in the output queue should always be served at rate equal to the output channel capacity. However, in soft multiplexing, since service capacity can be saved up, a soft server has the flexibility to control when to consume the resource and output a packet. In RPTP system, p_i^k should not depart until the time t at which $V(t) = F_i^k$. To keep track of these departure events, we use similar technique as keeping track of the departure events in the fictitious system. Let F_{min} be the minimum VFT in the output queue and $Next(t)$ be the prediction time of the next departure event in the RPTP system at time t, assuming that there is no arrival in the interval $(t, Next(t))$. Then, $Next(t)$ can be found by

$$F_{min} = V(t) + \frac{Next(t) - t}{\sum_{j \in B(t)} \rho_j} \rho \tag{5.10}$$

Similarly, we can set up an interrupt at time $Next(t)$ and recalculate $Next(t)$ if there is an arrival in $(t, Next(t))$.

Note that the algorithm we use is analogous to the one in [4], especially in the definition of V(t), the updating of V(t) and the calculation of $\bar{F}_i^k$. One drawback of this algorithm, as mentioned in [6], is that it needs to keep track of the set $B(t)$ to update $V(t)$. As there are no real leaky buckets and fictitious queues, we need to simulate the departure events in the fictitious system(which can be viewed as a FFQ system) to keep track of the states of the sessions. Another problem is that the departure events in the fictitious system may occur simultaneously (or very close to each other). It is possible that the number of departure events in an arbitrary short period will approach the total number of sessions and each of them may trigger the updating of $V(t)$. Consider the situation in which a packet arrives just after n simultaneous departure events have occurred. Then its VFT cannot be calculated until n updating of $V(t)$ have been finished. Because it is possible that this packet may require immediate departure, there will be

errors when using this algorithm to implement RPTP. Moreover, in high-speed networks, the number of sessions may approach several hundreds, the potential error will be even larger.

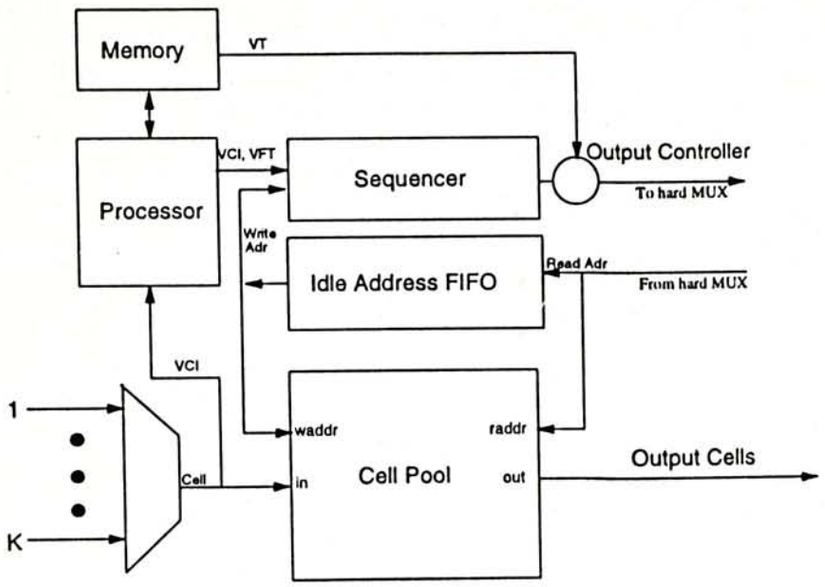

Figure 5.1: Hardware structure of virtual time implementation.

Fig. 5.1 shows the hardware structure of a virtual time implementation. The maximum number of cells arriving to the multiplexer in one time slot is limited by the switch design. Suppose at most K cells may arrive in one time slot. These cells are multiplexed and written one-by-one into the cell pool at the idle addresses supplied by idle address FIFO. Simultaneously, the VCI and VPI of each cell is forwarded to the processor which takes the responsibility of maintaining the virtual clock and calculating the VFT of each arriving cell. The sequencer works as an output queue and keeps the records (VFT and address) in the order of their VFT's. In [21], Chao has proposed an architecture of the sequencer which can insert a record into the queue in real time. The output controller may initiate a packet transmission with reference to the memory content, such

as virtual time and $B(t)$.

Obviously, the bottleneck of this design is in the processor. As we have mentioned, error may occur when calculation of VFT is deferred. The potential error increases when the number of sessions increases. Therefore, the number of sessions supported is limited. Moreover, error will also occur when n packets arrive simultaneously. However, an important advantage of this design is that the hardware complexity will not grow when number of sessions increases.

5.2 Leaky Bucket Implementation

Another implementation of RPTP is depicted in fig. 5.2. The main difference from the previous design is that we use a rate-adjustable clock to represent the virtual time physically. N leaky buckets are use to control the access of the sessions to the output channel. The rate-adjustable clock will generate clock pulses at a rate controlled by the clock rate controller (CRC). Normally, clock pulses are generated at a basic clock rate X. Clock pulses are generated at a higher rate if control signal is received from the CRC. The CRC in turn receives signals from the N leaky buckets. Chao [22] has suggested an hardware implementation of credit manager which consists of N leaky buckets. One modification required is that each leaky bucket needs to issue a state change signal to the CRC when the bucket becomes full or not full. Each leaky bucket is initialized with two parameters : 1) the bucket size, δ_i 2) the number of clock clicks between two generations of tokens, γ_i. Note that γ_i is derived from the token generation rate, ρ_i and the basic clock rate, X.

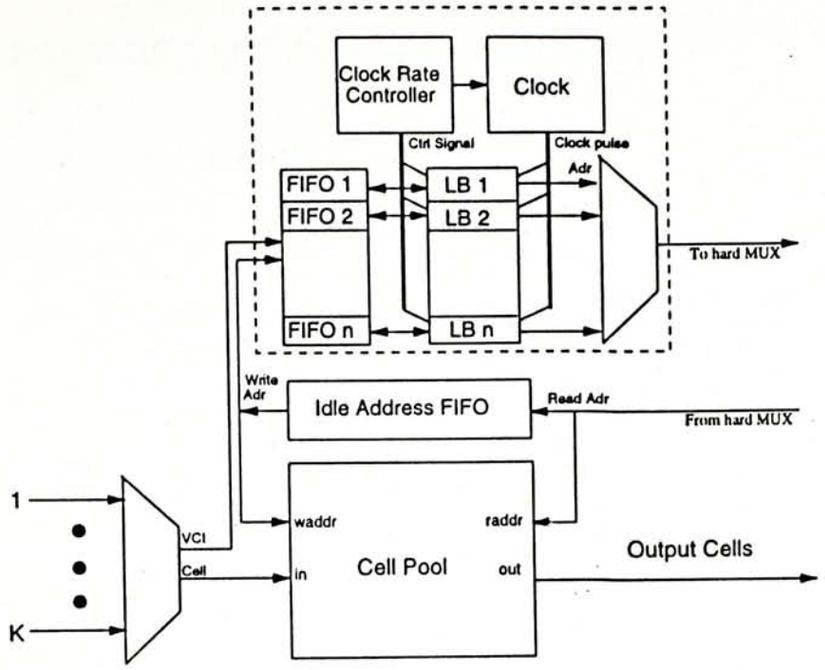

Figure 5.2: Leaky Bucket Implementation of RPTP.

According to the VCI, cells from different VC's will be served by the corresponding leaky buckets. State change signals (full or not full) will be sent from the leaky buckets to the CRC which will maintain the set of busy sessions $B(t)$. Then CRC will send control signals to the clock and set the clock rate to $\frac{\rho}{\sum_{i \in B(t)} \rho_i} X$. By doing so, when all sessions are busy, the clock will advance with the basic clock rate such that each session may have token generation rate ρ_i. On the other hand, when some buckets become full, the clock will advance faster and thus the remaining busy sessions may have a higher token generation rate.

An advantage of this parallel structure is that the error introduced is negligible. However, a FIFO and a leaky bucket is required for each session. Thus the hardware complexity is proportional to the number of sessions supported.

5.3 Summary of This Chapter

1. Virtual time implementation

a. Every arriving packet is associated with a time stamp, the virtual finishing time (VFT) of the packet in the RPTP system. A packet will depart the system when its time stamp is large than or equal to the current virtual time.

b. An advantage of virtual time implementation is that hardware complexity will not grow when number of sessions increases.

c. Error may occur when calculation of VFT is deferred by updating of virtual time, or when multiple packets arrive simultaneously.

2. Leaky-bucket implementation

a. The virtual time here is maintained physically by using a rate-adjustable clock. The N leaky buckets controlling the output of the sessions may have variable token generation rate depending on the set of busy sessions.

b. An advantage of leaky-bucket implementation is that RPTP can be implemented accurately.

c. Hardware complexity if proportional to the number of sessions supported.

3. Although the concept of virtual time implementation is borrowed from Parekh, we have shown that previous implementations for hard multiplexers can be modified for soft-multiplexer implementations via the fictitious system.

Chapter 6

Conclusion

In this research, we have established a framework for soft multiplexing (i.e. multiplexing on to a (δ, ρ) channel), which may occur in two scenarios : (1) multiplexing of VC's on to a (δ, ρ) VP in an ATM network; (2) multiplexing of sessions on to a permanent virtual connection (PVC) in a virtual private network (VPN). To motivate this study, from the network level point-of-view, we have shown by simulations that (δ, ρ) VP's can yield better overall statistical performance in ATM networks than deterministic VP's can. Assuming the FCFS scheduling scheme, this study, which can be found in chapter 2, focuses on the overall delay and backlog distributions of the *aggregate* traffic of all sessions being multiplexed on to the same physical link.

The performance (i.e. QOS) of individual sessions is important from the network-user's standpoint; and it depends on the scheduling scheme used. Chapter 3 investigates the effects of different scheduling schemes on individual sessions in a soft multiplexer. We have compared three schemes, namely first come first serve (FCFS), fixed-resource allocation and excess token passing. These

comparisons show that there is a trade-off between the average performance of all sessions and the performance guarantee to individual sessions. Among the three schemes, FCFS has the best overall average performance but the worst resource-guarantee (or performance guarantee) capability; fixed-resource allocation, while being able to guarantee performance of individual sessions, suffers from poor overall average performance, because the transmission resources are not shared at all, so that busy sessions cannot make use of unused resources of idle sessions. In the excess token passing scheme, guaranteed resources of idle sessions are available to the busy sessions. Thus it can provide strong guarantee and good (not the best) overall average performance.

Chapter 2 and 3 concern statistical performance. Besides the statistical performance, whether a soft multiplexer can provide worst-case guarantee (such as a bound on delay) is also an essential issue for it to be used in integrated-service networks. Chapter 4 analyzes the worst-case delay and backlog for one of many excess token passing schemes called the rate proportional token passing (RPTP). A fictitious system, which provides a mapping between soft-multiplexing and traditional hard-multiplexing schemes (which assume constant-bit rate output channel), has been constructed. Via this fictitious system, we have found that a lot of issues in soft multiplexing can be understood by borrowing results from previous work on hard multiplexing. For example, the hard-multiplexing scheme in the corresponding fictitious system of RPTP is actually the fluid-flow fair queueing(FFQ). This allows us to derive the results of the RPTP system from Parekh's work. Specifically, we have proved that when the sources are controlled by leaky buckets before they are multiplexed, a RPTP multiplexer can provide deterministic delay and backlog guarantee. The scenario under which

this maximum delay/backlog occurs has been derived.

Finally, to investigate the feasibility and practicality of RPTP, we have considered two possible implementations of it, namely virtual time implementation and leaky-bucket implementation. In particular, we have shown how previous implementations for hard multiplexer can be modified for soft-multiplexer implementation.

Several areas require further investigations. First of all, for the worst-case delay/backlog in a single RPTP node, although we know that they occur when all the sessions start to be greedy at the beginning of a system busy period, explicit calculations of them are complicated when the number of sessions is large (see Section 4.5). When using this worst case delay/backlog in bandwidth allocation, it is necessary to find an efficient way to calculate them. The second area is to extend the delay/backlog bound to multiple-node case, or in other words, to bound the end-to-end delay/backlog in ATM networks. One interesting feature that arises in the end-to-end performance analysis of soft-multiplexing is that the traffic characteristics on a VP may be changed when this VP is being hard-multiplexed on to a physical link. Therefore, the output traffic characteristic of the upstream soft multiplexer is not equal to the input traffic characteristic of the downstream soft multiplexer. Some preliminary results of this topic are available in the appendix. Third, only several soft-multiplexing schemes have been considered in this thesis. Many other schemes are possible. It will be interesting to explore these other soft-multiplexing schemes, using the methodologies established in this work.

Appendix A

End-to-end Delay/Backlog Bound in ATM Networks

So far, we have only considered the delay and backlog in a single soft multiplexer. In this appendix, we extend our early result in bounding the end-to-end queueing delay and backlog of a VC in ATM network.

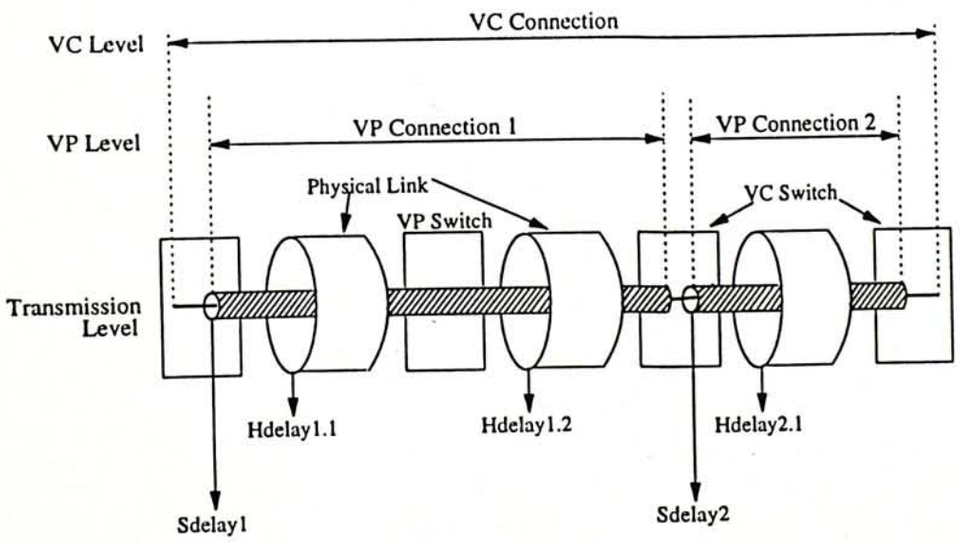

Figure A.1: Transport Architecture and Decomposition of QOS in ATM Networks

To define our focus in ATM networks, consider Fig. A.1. End-to-end queueing delay of VC i, $VCdelay_i$, can be calculated as follow [14]:

$$VCdelay_i = \sum_{j=1}^{nvp_i} \{Sdelay_j + Hdelay_j\} \quad (A.1)$$

where nvp_i is the number of VP's traversed by VC i, $Sdelay_j$ is the queueing delay when the VC is soft multiplexed onto the jth VP and $Hdelay_j$ is the queueing delay inside the jth VP due to a series hard multiplexers. $Hdelay_j$ can be further decomposed into $Hdelay_{j.1}, \cdots, Hdelay_{j.m}$ where m is the number of physical links traversed by the corresponding VP.

Therefore, the end-to-end queueing delay can be separated into two components: delay in the soft multiplexers,$\sum_{j=1}^{nvp_i} Sdelay_j$ and those in the VP's, $\sum_{j=1}^{nvp_i} Hdelay_j$. A similar equation also exists for end-to-end backlog. To bound the end-to-end queueing delay/backlog, both components must be bounded. By using an appropriate hard-multiplexing discipline to multiplex VP's onto physical links, the latter component can be bounded given that the input traffic of each VP conforms to the pre-agreed leaky-bucket parameters. One hard-multiplexing discipline providing such guarantee is packet-by-packet fair queueing(PFQ) which is the implementation version of fluid-flow fair queueing(FFQ) in actual packet scenario. Parekh and Gallager have proven that when the sources are controlled by leaky bucket, the queueing delay and backlog over multiple nodes are bounded. Moreover, the departure time of a packet in the PFQ system will not be larger than the departure time in the FFQ system by 1 packet time. Thus, we can confine our attention to bound the first component, delay/backlog within the soft multiplexers.

The main result in this section is that the end-to-end queueing delay/backlog is bounded, if the following three conditions are satisfied : (1) all VC's form

a acyclic topology; (2) each VC is leaky-bucket-controlled; (3) each VP has guaranteed delay and guaranteed backlog.

Let us clarify the notation before going on. Define $M_i^{(k)}$ to be the kth soft multiplexer on the path of VC i and $I(m)$ to be the set of VC's on soft multiplexer m. Denote the amount of VC i traffic arriving to $M_i^{(k)}$ in the interval $(\tau, t]$ as $A_i^{(k)}(\tau, t)$. Thus $A_i^{(1)}$ indicates the traffic of VC i entering the network. If each VC is leaky-bucket-controlled, then for all i,

$$A_i^{(1)}(\tau, t) \leq \delta_i^{(1)} + \rho_i^{(1)}(t - \tau)$$

Let $S_i^{(k)}(\tau, t)$ be the amount of VC i traffic served by $M_i^{(k)}$ in the interval $(\tau, t]$. Normally, when doing this kind of analysis, people assume $A_i^{(k)} = S_i^{(k-1)}$. However, this assumption is invalid in ATM networks with the VC/VP hierarchy because, before entering $M_i^{(k)}$, output traffic from $M_i^{(k-1)}$ may go through a VP after which traffic characteristics will be changed by a series of hard multiplexing.

Denote $VP_i^{(k)}$ the kth VP on the path of VC i. Moreover, for all i and $k = 1, \cdots, nvp_i$, $VP_i^{(k)}$ has delay bound $VPD_i^{(k)}$ and backlog bound $VPQ_i^{(k)}$. To bound the end-to-end delay/backlog of a VC, we use the additive method described as follow:

(1) For each VC i in the network and $k = 1, \cdots, nvp_i$, we find a minimum value, $\delta_i^{(k)}$, such that $A_i^{(k)}$ conforms to $(\delta_i^{(k)}, \rho_i^{(1)})$.

(2) For each $M_i^{(k)}$, $k = 1, \cdots, nvp_i$, calculate the worst case delay, $D_i^{(k)}$ and worst case backlog, $Q_i^{(k)}$, of VC i based on Theorem 2. That is to assume all the VC's in $I(M_i^{(k)})$ starts to be greedy together based on the worst case traffic characteristics obtained in (1).

(3) Then the end-to-end queueing delay of VC i is bounded by $\sum_{k=1}^{nvp_i}(D_i^{(k)} + VPD_i^{(k)})$. And the end-to-end backlog of VC i is bounded by $\sum_{k=1}^{nvp_i}(Q_i^{(k)} + VPQ_i^{(k)})$.

Therefore, if we can find $\delta_i^{(k)}$ to characterize the internal arrival traffic, the end-to-end delay and backlog bounds can be easily obtained by (2) and (3).

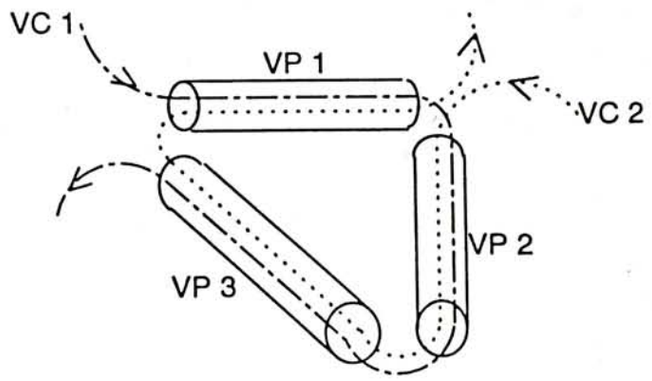

Figure A.2: A cyclic VC topology.

Although the route of a single VC is acyclic, the union of the routes of several VC's may form a cycle just like what Fig. A.2 shows. In this example, while $A_1^{(2)}$ depends on $A_2^{(3)}$, $A_2^{(3)}$ depends on $A_1^{(2)}$ also. This bi-directional dependence arouses a feedback effect which complicates the characterization process of the internal arrival traffic. Therefore, in this thesis, we only consider acyclic VC topology. In the following, we describe a method to characterize the internal arrival traffic of a acyclic VC topology. Consider a particular VC i. Assume that $A_i^{(k)}$ conforms to $(\delta_i^{(k)}, \rho_i^{(1)})$ and all other arrival traffic entering $M_i^{(k)}$ has already been characterized. By using the technique mentioned in Section 5.1, we can find the smallest value of $\hat{\delta}_i^{(k),out}$ such that $\hat{S}_i^{(k)}$ conforms to $(\hat{\delta}_i^{(k),out}, \rho_i^{(1)})$. Recall that $\hat{S}_i^{(k)}$ is the service function under the all-greedy arrival pattern. Moreover,

it can be shown that,

$$\hat{\delta}_i^{(k),out} \leq \delta_i^{(k),out}$$

for any arrival pattern. This means that $S_i^{(k)}$ always conforms to $(\hat{\delta}_i^{(k),out}, \rho_i^{(1)})$. Before arriving at $M_i^{(k+1)}$, traffic of VC i will go through $VP_i^{(k)}$ after which its burstiness may increase because of a series of hard multiplexing, To characterize $A_i^{(k+1)}$, we make use of a theorem from Cruz [18]. For the proof of this theorem, please refer to [18].

THEOREM 3: If traffic with rate function $R_{in} \sim (\delta, \rho)$ passes through a element with delay bound D, then the output traffic from this element, $R_{out} \sim (\delta + D\rho, \rho)$.

According to Theorem 3, the output traffic of VC i from $VP_i^{(k)}$, i.e. $A_i^{(k+1)}$, must conform to $(\hat{\delta}_i^{(k),out} + VPD_i^{(k)}\rho_i^{(1)}, \rho_i^{(1)})$. Because we know that, for all i, $A_i^{(1)}$ conforms to $(\delta_i^{(1)}, \rho_i^{(1)})$, we can characterize the internal arrival traffic, $A_i^{(k)}$, for $k = 1, \cdots, nvp_i$, by applying the procedure mentioned above repeatedly if the VC topology is acyclic.

Note that in this end-to-end delay bound, we have considered each of $M_i^{(k)}$, $k = 1, \cdots, nvp_i$, in isolation. Moreover, if two VC i and j are both served by the same node n, just before they contend at another node m, then it may not be possible for both of them to be simultaneously greedy at node m. Thus, the calculated worst case delay and backlog are loose bounds. Further investigation is needed for tighter bounds. Moreover, characterization of internal traffic for cyclic VC topology should also be studied.

Bibliography

[1] C. W. Lai and Soung C. Liew, "A Framework for Statistical Multiplexing onto a Variable-Bit Rate Output Channel," *Proc. IEEE INFOCOM'95*, pp 447-454.

[2] E. Hahne, "Round robin scheduling for fair flow control, " *Ph.D. thesis, Dept. Elect. Eng. and Comput. Sci., M.I.T.*, Dec. 1986.

[3] A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm." *Proc. SIGCOMM'89*, pp.1-12.

[4] 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. on Networking*, Vol.1, No.3, June 1993, pp.344-357.

[5] A. K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in integrated Services Networks : The Multiple Node Case," *Proc. IEEE INFOCOM'93*, Mar. 1993, pp.521-530.

[6] S. J. Golestani "A Self-Clocked Fair Queueing Scheme for Broadband Applications" *Proc. INFOCOM'94*, pp.636-646.

[7] L. Zhang, "Virtual clock : A new traffic control algorithm for packet switching" *ACM Transactions on Computer Systems*, 9(2):101-124, may 1991.

[8] S. J. Golestani, "Congestion-Free Transmission of Real-Time Traffic in Packet Networks" *Proc. IEEE INFOCOM'90*, San Francisco, Jun. 1990, pp.527-536.

[9] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management" *Proc. ACM SIGCOMM'90*, Vol.20, No.4, Sept. 1990, pp.8-18.

[10] S. J. Golestani, "Duration-Limited Statistical Multiplexing of Delay-Sensitive Traffic in Packet Networks" *Proc. IEEE INFOCOM'91*, Bal Harbor, Fl., Apr. 1991, pp.323-332.

[11] J.B. Nagle, "On Packet Switches with Infinite Storage" *IEEE Trans on Commun.*, Vol.35, No.4, Apr. 1987, pp.435-438.

[12] M. Kawarasaki, B. Jabbari, "B-ISDN Architecture and Protocol" *IEEE J. select. Areas Commun.*, Vol.9, No.9, Dec. 1991, pp.1405-1415.

[13] K. Sato, S. Ohta, I. Tokizawa, "Broad-Band ATM Network Architecture Based on Virtual Paths" *IEEE Trans on Commun.*, Vol.38, No.8, Aug. 1990, pp.1212-1222.

[14] Y. Katsube, T. Kodama, "QOS Control in an ATM Network considering Virtual Path Concept" *IEEE GLOBECOM'90*, 1990, pp.1104-1110.

[15] J. S. Turner, "New Direction in Communications or Which Way to the Information Age?" *Proc. Int. Zurich Sem. Digit. Commun. '86*, pp.25-32.

[16] M. Butto, E. Cavallero and A. Tonietti, "Effectiveness of the Leaky Bucket Policing Mechanism," *IEEE J. select. Areas Commun.*, Vol.9, No.3, Apr. 1991, pp.335-342.

[17] N. Yin, M. G. Hluchyj, "Analysis of the Leaky Bucket Algorithm for On-Off Data Source" *Proc. GLOBECOM'91*, pp.254-260.

[18] R. L. Cruz, "A Calculus for Network Delay, Part 1 : Network Elements in Isolation" *IEEE Tran. on Information Theory*, Vol.37, No.1, Jan 1991, pp.114-131.

[19] R. L. Cruz, "A Calculus for Network Delay, Part II : Network Analysis" *IEEE Tran. on Information Theory*, Vol.37, No.1, Jan. 1991, pp.132-141.

[20] L. Kleirock, *Queueing System*, Vol.2, Computer Application, John Wiley, pp.1141-1147.

[21] H. J. Chao, "A VLSI Sequencer Chip for ATM Traffic Shaper and Queue Manager" *IEEE Journal of Solid-State Circuits*, Vol. 27, No.11, Nov. 1992.

[22] H. J. Chao, "Design of Leaky Bucket Access Control Schemes in ATM Networks" *IEEE International Conference on Communications*, pp.180–187.