Spring 5-31-2001

Design of traffic shaper / scheduler for packet switches and DiffServ networks : algorithms and architectures

Surong Zeng

New Jersey Institute of Technology

ABSTRACT

DESIGN OF TRAFFIC SHAPER/SCHEDULER FOR PACKET SWITCHES AND DIFFSERV NETWORKS: ALGORITHM AND ARCHITECTURE

by
Surong Zeng

The convergence of communications, information, commerce and computing are creating a significant demand and opportunity for multimedia and multi-class communication services. In such environments, controlling the network behavior and guaranteeing the user's quality of service is required. A flexible hierarchical sorting architecture which can function either as a traffic shaper or a scheduler according to the requirement of the traffic load is presented to meet the requirement. The core structure can be implemented as a hierarchical traffic shaper which can support a large number of connections with a wide variety of rates and burstiness without the loss of the granularity in cells' conforming departure time. The hierarchical traffic shaper can implement the exact sorting scheme with a substantial reduced memory size by using two stages of timing queues, and with substantial reduction in complexity, without introducing any sorting inaccuracy.

By setting a suitable threshold to the length of the departure queue and using a lookahead algorithm, the core structure can be converted to a hierarchical rate-adaptive scheduler. Based on the traffic load, it can work as an exact sorting traffic shaper or a Generic Cell Rate Algorithm (GCRA) scheduler. Such a rate-adaptive scheduler can reduce the Cell Transfer Delay and the Maximum Memory Occupancy greatly while keeping the fairness in the bandwidth assignment which is the inherent characteristic of GCRA. By introducing a best-effort queue to accommodate best-effort traffic, the hierarchical sorting architecture can be changed to a near work-conserving scheduler. It assigns remaining bandwidth to the best-effort traffic so that it improves the utilization of the outlink while it guarantees the quality of service requirements of those services which require quality of service guarantees. The inherent flexibility of the hierarchical sorting architecture combined with intelligent algorithms determines its multiple functions. Its implementation not only can manage buffer and bandwidth resources effectively, but also does not require no more than off-the-shelf hardware technology.

The correlation of the extra shaping delay and the rate of the connections is revealed, and an improved fair traffic shaping algorithm, Departure Event Driven plus Completing Service Time Resorting algorithm, is presented. The proposed algorithm introduces a resorting process into Departure Event Driven Traffic Shaping Algorithm to resolve the contention of multiple cells which are all eligible for transmission in the traffic shaper. By using the resorting process based on each connection's rate, better fairness and flexibility in the bandwidth assignment for connections with wide range of rates can be given.

A Dual Level Leaky Bucket Traffic Shaper(DLLBTS) architecture is proposed to be implemented at the edge nodes of Differentiated Services Networks in order to facilitate the quality of service management process. The proposed architecture can guarantee not only the class-based Service Level Agreement, but also the fair resource sharing among flows belonging to the same class. A simplified DLLBTS architecture is also given, which can achieve the goals of DLLBTS while maintain a very low implementation complexity so that it can be implemented with the current VLSI technology.

In summary, the shaping and scheduling algorithms in the high speed packet switches and DiffServ networks are studied, and the intelligent implementation schemes are proposed for them.

DESIGN OF TRAFFIC SHAPER/SCHEDULER FOR PACKET SWITCHES AND DIFFSERV NETWORKS: ALGORITHM AND ARCHITECTURE

by
Surong Zeng

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

Department of Electrical and Computer Engineering

May 2001

APPROVAL PAGE

DESIGN OF TRAFFIC SHAPER/SCHEDULER FOR PACKET SWITCHES AND DIFFSERV NETWORKS: ALGORITHM AND ARCHITECTURE

Surong Zeng

Dr. Necdet Uzun, Dissertation Co-Advisor Date
Chief Architect, AuroraNetics, Inc.

Dr. Symeon Papavassiliou, Dissertation Co-Advisor Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Nirwan Ansari, Committee Member Date
Professor of Electrical and Computer Engineering, NJIT

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

Dr. Haldun Hadimioglu, Committee Member Date
Associate Professor of Computer and Information Science,
Polytechnic University, Brooklyn, NY

BIOGRAPHICAL SKETCH

Author:	Surong Zeng
Degree:	Doctor of Philosophy
Date:	May 2001

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ 2001

- Master of Science in Electrical Engineering,
  University of Electronic Science and Technology of China,
  Chengdu, Sichuan, P.R.China, 1998

- Bachelor of Science in Electrical Engineering,
  University of Electronic Science and Technology of China,
  Chengdu, Sichuan, P.R.China, 1995

Major: Electrical Engineering

Publications and Presentations:

S. Zeng, N. Uzun,
"A Hierarchical Traffic Shaper for Packet Switches",
Proc. of IEEE GLOBECOM'99, pp. 1655-1659,
Rio de Janeiro, Brazil, Dec. 1999.

S. Zeng, N. Uzun, T. Mehmet, and N. Gogate,
"A Novel Hierarchical Rate-Adaptive Scheduler for High Speed Packet Switches",
Networld+Interop'2000, Las Vegas, NV, May 2000.

S. Zeng, N. Uzun,
"A Novel Hierarchical Traffic Shaper/Scheduler for High Speed Packet Switches",
submitted to *IEEE/ACM Transaction on Networking*, March 2000.

S. Zeng, N. Uzun, and S. Papavassiliou,
"An Improved Fair Traffic Shaping Algorithm for High Speed Packet Switches",
accepted by *Proc. of 2001 IEEE Workshop on High Performance Switching and Routing*, Dallas, TX, May 2001.

S. Zeng, N. Uzun, and S. Papavassiliou,
"A Dual Level Leaky Bucket Traffic Shaper Architecture for DiffServ Networks",
accepted by *Proc. of 2001 IEEE Workshop on High Performance Switching and Routing*, Dallas, TX, May 2001.

S. Zeng, N. Uzun, and S. Papavassiliou,
"A Dual Level Access Control Architecture for DiffServ Networks"
submitted to *Journal of Network and Systems Management*, April 2001.

This work is dedicated to my beloved family

ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude to my advisors and mentors, Dr. Necdet Uzun and Dr. Symeon Papavassiliou. I would like to thank Dr. Uzun, an excellent advisor and a good friend, for introducing me to this exciting field of research; for his constant support, guidance and encouragement in every step of my way; and for his inspiration and valuable discussions. I appreciate Dr. Papavassiliou for his excellent suggestions; and for the time he has put into help me to make the dissertation complete.

I would like to thank my dissertation committee, Dr. Nirwan Ansari, Dr. Sirin Tekinay, and Dr. Haldun Hadimioglu (Brooklyn Polytechnic), for their being interested in this work and stimulating valuable discussions and suggestions.

I would like to thank all fellow researchers at NJCMR for all the help they have offered during my doctoral research. I would especially like to thank Feihong Chen and Jie Yang for patiently discussing with me, and giving me valuable suggestions. I would also like to thank Qiang Peng, Bin He, Xufei Wei, Minyi Zhao, Xiaodong Cai for their continuous help and advice since the first day I joined NJIT. I would like to thank all my friends at NJCMR for making my stay here enjoyable.

Words cannot express the gratitude I felt toward my parents, whose support provides the foundation upon which this work was built.

TABLE OF CONTENTS

Chapter Page

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

1.1 Background

Broadband networks must accommodate integrated services with a very wide range of Quality of Service (QoS) requirements, such as voice, video, and data. In order to utilize network resources effectively while providing a broad range of satisfactory QoS guarantees to all users on the network, traffic regulation at the network edge is necessary. Connections are monitored by network operators based on the traffic contract established during admission control [15]. If a connection does not conform to the traffic contract, a Usage Parameter Control (UPC) device at the network edge can discard the violating cells or tag them to a lower priority. It also can employ traffic shaping to delay non-conforming cells until they are compliant with conditions described by the traffic contract [45, 61, 88].

The widely used traffic shaping and scheduling algorithm is the Generic Cell Rate Algorithm (GCRA), which is also known as leaky bucket algorithm [7, 12, 39, 71, 79]. The GCRA is used to judge whether an incoming cell conforms to the negotiated QoS parameters, and calculate its earliest conformance time. The QoS parameters for each service are a set of values that pertain to delay and loss of the transferred cells. These values are negotiated between the user and the network, and the network promises to guarantee the QoS requirements of the user's service only when the user conforms to these values. The user has to specify the traffic parameters of the source so that the network can estimate the bandwidth correctly. These traffic parameters may include Peak Cell Rate (PCR), Cell Delay Variation Tolerance (CDVT), Sustained Cell Rate (SCR), and Burst Tolerance (BT) [34]. A different set of traffic parameters is specified for each service category that the network provides.

In the design of a traffic shaper, three essential attributes should be taken into consideration: (1) low system delay for incoming cells; (2) simplicity and low cost in the implementation of sorting time stamps in order to select the cell with the minimum conformance time; and (3) fairness of bandwidth assignment. Generally, the delay, and the implementation complexity are the most important criteria for a system design [82]. The implementation complexity and cost are dominated by the complexity of sorting and storing time stamps [23]. Based on the GCRA, many implementations of traffic shapers are proposed, such as direct exact sorting scheme, rate-based grouping scheme, etc.

1.2 Outline of the Dissertation

A new traffic shaper architecture with hierarchical timing queues is proposed in this dissertation. In this new architecture, two stages of timing queues are used to implement the exact sorting algorithm so that this shaper architecture can substantially reduce the number of subqueues needed to store time stamps. Meanwhile, the exact sorting algorithm is combined with the UPC compliant departure event driven traffic shaping (DEDTS) algorithm [15], so that there is at most one cell for every connection in the sorting unit at any given time. Consequently, sorting unit complexity can be reduced. In contrast to various existing architectures for implementing a traffic shaper, the proposed implementation can reduce the cost and complexity greatly by reducing memory size substantially and eliminating complex arbitration logic of a scheduler.

This hierarchical traffic shaper can be implemented as a hierarchical rate-adaptive scheduler by using a rate-based lookahead algorithm. When the traffic load is light, this hierarchical rate-adaptive scheduler can work as a GCRA scheduler to reduce the Cell Transfer Delay (CTD) and Maximum Memory Occupancy (MMO) in comparison with those of the traffic shaper. When the traffic load is heavy, it performs the shaping function in order to avoid the congestion of the outlink.

This hierarchical sorting architecture can also be implemented as a near work-conserving scheduler. Generally, in a high speed packet switch hardware design, a memory limit and a cell loss ratio limit should be given. Under these two conditions, the maximum accepted traffic which requires QoS guarantees and the maximum outlink utilization are set. Under the same conditions, a best-effort queue is introduced to accommodate the best-effort traffic, and the remaining bandwidth is assigned to it. Such a scheduler not only improves the outlink utilization, but also relaxes the Connection Admission Control(CAC) constraints.

An improved fair traffic shaping algorithm is also proposed in this dissertation. It modifies the DEDTS algorithm by adding a process to resort the Completing Service Time stamps (CT) of all conforming cells. The CT is the sum of the conforming Departure Time (DT) and the service time of its associated connection. The service time of a connections is the reciprocal of the rate of the connection. By resorting the CT stamps, conforming cells can be prioritized based on the rates of the connections, and the available bandwidth can assigned more fairly so that this algorithm reduces the shaping delay of high rate connections greatly. This algorithm can be implemented by modifying the Hierarchical Timing Queues Traffic Shaper architecture easily.

Currently, the Internet can not support Quality of Service (QoS) requirements. With the growth of the Internet, more and more applications demand QoS support from the networks. To meet these demands, the Integrated Services architecture was proposed first by Internet Engineering Task Force (IETF). In the Integrated Services (IntServ) networks, all QoS mechanisms are flow-based. However, there are generally thousands, even ten or hundred thousands of flows in the network at the same time. The state information and processing capability are proportional to the number of flows, therefore, IntServ is definitely unscalable to the Internet. In order to resolve the scalability problem while keeping the QoS support function, Differentiated Services (DiffServ) architecture is proposed [4, 91].

In the DiffServ networks, the traffic conditioner is one of the most important part, and the shaper is one of its components. In the IETF draft, a rate adaptive shaper for DiffServ networks was proposed [6]. However, it doesn't resolve the fair resource sharing problem for flows in the same class. In this dissertation, a Dual Level Leaky Bucket Traffic Shaper (DLLBTS) architecture is proposed for the edge nodes of DiffServ networks. In DLLBTS architecture, there are two levels of leaky buckets. The first level is a set of flow-based leaky buckets. It can shape flows based on the traffic profile of each flow so that it can guarantee that all flows can fairly share the network resources in the same class when they merge into one class. The second one is a set of class-based leaky buckets same as all other traffic shapers for interior nodes of the DiffServ networks. They can guarantee the aggregate behavior of a class conforming to the Service Level Agreement (SLA). However, two sets of leaky bucket traffic shaper need two sets of sorting units, and generally, the sorting unit dominates the complexity of the implementation of a traffic shaper. It is an expensive solution to implement two sets of complex sorting units in a traffic shaper. A simplified DLLBTS is proposed, which combines the two levels of leaky buckets to one stage so that only one set of sorting unit is needed. The simulation results show that the simplified DLLBTS can reach the goal of DLLBTS while keeping the low implementation complexity.

The dissertation is organized as follows. The second chapter reviews the related work, including traffic management, traffic parameters, event-driven traffic shaping algorithms, and QoS in Internet. The third chapter describes the details of the proposed hierarchical traffic shaper architecture, and compares it with two other architectures. It shows that the new architecture has lower implementation complexity and cost. In this chapter, the performance evaluation for the hierarchical traffic shaper is given, and it shows that the simplified implementation performs as well as the exact sorting scheme. The fourth chapter presents the rate-adaptive scheduler implemented by hierarchical sorting units with a lookahead algorithm, and gives the performance evaluation. In the same chapter, the hierarchical near work-conserving scheduler with best effort traffic is also given. It is implemented by adding a best-effort queue to accommodate the best effort traffic, and it shows that the proposed technique improves the outlink utilization without increasing memory occupancy and cell loss ratio for service with QoS requirements. The fifth chapter depicts the improved fair traffic shaping algorithm and its implementation. The performance evaluation shows that this proposed traffic shaping algorithm can give better fairness performance in the bandwidth allocation. The sixth chapter proposes DLLBTS for the edge node of DiffServ networks, and a simplified DLLBTS architecture is proposed for implementation purposes. The last chapter draws conclusions.

CHAPTER 2
RELATED WORK OVERVIEW

The network is becoming ubiquitous, and more and more applications require the network to support high quality of service which generally has to consume much more network resources. Although the network capacity is increasing at an astounding rate, users still suffer from the congested network, and with the development of the high-bandwidth requirement applications, this situation will not be alleviated in the near future [4]. In order to enhance the ability of a network to offer high-quality service guarantees while simultaneously making efficient use of raw network resources, sophisticated QoS mechanisms have to be adopted. QoS mechanisms are a set of technologies that enable network administrators to reduce, even avoid the effects of congestion on application traffic by using network resources optimally, rather than by continually adding capacity [4]. The most common QoS mechanisms include scheduling, shaping, policing, admission control, intelligent routing etc. [91]. To meet appropriate differentiated QoS requirements, the ATM Forum proposed a framework for managing traffic and controlling congestion in ATM networks [34]. The framework consists of many QoS mechanisms, which will be discussed later.

In this chapter, the framework of the traffic management in ATM networks is described. Traffic regulations, including traffic descriptor, QoS parameters, and regulation algorithm based on these parameters are also discussed. Users specify the traffic parameters to describe their traffic in a quantitative means. According to the characteristics and QoS requirements of the traffic, service classes are defined in the ATM networks, which are discussed in the third section. Based on the traffic information and the current network state, administrators of the network can deploy CAC to make a connection decision. Once the connection is accepted, there is a traffic contract constructed. A traffic contract is a legal binding agreement between users and networks [44]. It is defined by the users' traffic descriptor, QoS requirement, and connection descriptor. In the whole duration of the communication, administrators employ Usage/Network Parameter Control (UPC/NPC) to monitor whether the behavior of the traffic conforms to the traffic contract or not. For the non-conforming traffic, there are many algorithms and regulations which can be deployed to regulate it. The widely used policing and shaping algorithm Generic Cell Rate Algorithm (GCRA) is explained, and the existing event-driven traffic shaping algorithms are discussed in the fourth section. Finally, the QoS in the Internet is discussed, and IntServ and DiffServ are described and compared in this chapter.

2.1 Traffic Management Framework

Traffic management is to protect the network and the end-system from congestion in order to achieve high network performance, provide high QoS guarantees, as well as promote the efficient use of the network resources [34, 81, 90].

In order to meet such objectives, a framework for managing and controlling traffic and congestion in ATM networks is proposed by ATM Forum [34]. It consists of the following QoS mechanisms:

- Connection Admission Control (CAC)
- Usage/Network Parameter Control (UPC/NPC) (traffic policing)
- Cell Loss Priority Control
- Traffic Shaping
- Network Resource Management (NRM)
- Feedback Controls
- Frame Discard (Memory Management)
- ABR Flow Control

- Other control functions

In this section, the functions in the QoS framework of ATM networks are discussed as the following, except the traffic shaping which will be studied in details in the later sections.

- Connection Admission Control (CAC):

  Connection admission control is the first line of defense for the network in protecting itself from excessive loads [81]. It is defined as the set of actions taken by the network at the call setup phase (or during a call renegotiation phase) to determine whether the connection can be accepted or not. Only when there are sufficient network resources to accommodate the new connection without impacting the performance of other existing connections, the new connection can be accepted. At the instance of the connection establishment, the traffic contract is formed by being negotiated with the user. During the whole period of the connection lifetime, the network resources will be deployed to satisfy the QoS requirements of the connection if its traffic complies with the traffic contract [34, 58, 81, 90].

  In ATM networks, the CAC is to determine whether a Virtual Channel Connection (VCC) or a Virtual Path Connection (VPC) can be accepted. In the multimedia application scenario, one call can require more than one connection, then the CAC procedures are performed for each VCC or VPC [90]. Some CAC algorithms and schemes are proposed in [31, 29, 37, 53, 75].

- Usage/Network Parameter Control (UPC/NPC)(traffic policing):

  The UPC and the NPC are defined as a set of actions taken by the network to monitor and control traffic conforming to the traffic contract at the User-Network Interface (UNI) and the Network-Network Interface (NNI), respectively. They are also known as traffic policing, which is one of the most important QoS mechanisms. The main role of the UPC/NPC is to protect network resources from malicious as well as unintentional misbehavior by detecting violations of negotiated parameters, and taking appropriate action, such as marking, dropping, or shaping [90]. Networks may use different UPC/NPC algorithm to police different service classes. Algorithms for the UPC/NPC are primarily based on the GCRA which will be discussed later in this chapter. Many traffic policing algorithms are proposed and studied in [8, 60, 28, 92].

- Cell Loss Priority Control:

  Cell loss priority control is a set of priority policy for accepted connections. The end system may base the policy to mark the traffic from different service categories different Cell Loss Priority (CLP). If the traffic is treated as significant, the network may selectively drop cells with low CLPs to protect the QoS objectives of cells with high CLPs [16, 19, 27, 34, 64].

  When a traffic is not conforming to its traffic contract, even its contract specifies a high CLP for it, it can be marked as low priority to protect other conforming traffic from performance degradation due to the misbehavior of some connection.

- Traffic Shaping:

  Traffic shaping mechanisms are used to modify the traffic characteristics in a desired manner [34]. They will be discussed in the later sections.

- Network Resource Management (NRM)

  The service classification mode makes the logical separation of connections according to service characteristics possible, while the Virtual Paths makes the cell scheduling and resource provisioning to be able to provide appropriate isolation and access to resources. VPCs play a key role in network resource management [34].

VP bandwidth control reduces the required VP bandwidth due to the statistical multiplexing characteristics, and provides fine bandwidth assignment granularity so that the bandwidth of a VP can be precisely tailored to meet the demand [90]. The details of the resource management can be found in [17, 52, 56, 77, 80, 85, 86].

- Frame Discard (Memory Management):

  In ATM networks, if a cell is discarded, it may lead to upper layer to discard the whole frame or packet. It is more effective to discard at the frame level rather than at the cell level. Frame discard may help avoid congestion collapse and can increase goodput [34].

  Partial Packet Discard (PPD) and Early Packet Discard (EPD) are two widely used frame discard schemes.

  - Partial Packet Discard (PPD): If a cell from a packet is dropped from a switch buffer, the subsequent cells in that packet are also discarded [81].
  - Early Packet Discard (EPD): When a switch buffer reaches a threshold level, but before it is actually required to discard any cells due to buffer overflow, and entire packet is dropped. Then, when the switch senses that congestion is beginning and that cell discard may soon be necessary, it begins to drop some packets with low priority so that it can guarantee the goodput of packets with higher priority to some extent [81].

  Many modified versions of these schemes are proposed and studied in [11, 22, 24, 33, 36, 51, 62].

- Feedback Controls:

Feedback control is defined as the set of actions taken by both the network and users to regulate the traffic submitted to the connections according to the feedback of the state of network elements. Details of feedback control can be found in [34, 46, 54, 55, 68].

2.2 Traffic Parameters and QoS Parameters

Traffic parameters describe the characteristics of a connection. They can be grouped into the source traffic descriptor and the connection traffic descriptor. The source traffic descriptor describes the intrinsic characteristics of a source. The connection traffic descriptor describes the traffic characteristics at standardized network interfaces [10, 34, 44].

- Source Traffic Descriptor

  The source traffic descriptor is a set of parameters which captures the intrinsic characteristics of a source. It is used to request a connection establishment by the source. The main parameters are Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), Maximum Burst Size (MBS), and Minimum Cell Rate (MCR). Types of source traffic descriptors are dependent on the ATM service categories, which are listed in Fig. 2.2. For the VBR services, PCR, SCR and MBS are all defined, but for CBR, only the PCR is defined. These parameters are explained as follows.

  1. Peak Cell Rate (PCR) : The PCR specifies the upper bound on the source rate. Enforcement of this bound by the UPC allows the network operator to allocate sufficient resources to ensure that the network performance objectives (e.g., for Cell Loss Ratio) can be achieved. The inverse of PCR is the peak emission interval of the connection: T_p, which is enforced by UPC in the real operation [34, 44].

2. Sustainable Cell Rate (SCR) : The SCR indicates the upper bound on the conforming average rate of an ATM connection, over time scales which are long relative to those for which the PCR is defined. Enforcement of this bound by the UPC could allow the network to allocate sufficient resources, less than those based on the PCR, and still ensure that the performance objectives can be achieved. Similarly, the inverse of SCR T_s is enforced by UPC in the real operation [34, 44].

3. Maximum Burst Size (MBS): The MBS is the maximum number of cells transmitted in PCR. MBS together with PCR and SCR may determine the Burst Tolerance (BT).

4. Minimum Cell Rate (MCR) : The MCR is the rate at which the source is always allowed to send. This is the minimum allocated bandwidth of a connection.

- Connection Traffic Descriptor

  The connection traffic descriptor specifies the traffic characteristics of an ATM connection. The connection traffic descriptor includes the source traffic descriptor, the Cell Delay Variation Tolerance (CDVT), Burst Tolerance (BT), and the conformance definition that is used to specify the conforming cells of the connection.

  1. Cell Delay Variation Tolerance (CDVT): ATM layer functions (e.g., cell multiplexing), OAM cells, and other physical layer overhead may alter the traffic characteristics of connections by introducing Cell Delay Variation. Consequently, the peak emission interval T_p may be affected, which may lead to cell clumping phenomenon. The upper bound on the "clumping" measure is the CDVT. CDVT is defined in relation to the PCR according to the GCRA denoted by GCRA(T_p, CDVT).

2. Burst Tolerance (BT): BT also specifies the upper bound of cell clumping phenomenon with respect to the sustainable emission interval T_s. Consequently, BT is defined in relation to the SCR according to GCRA(T_s, BT).

3. Conformance Definition: The conformance of cells of a connection is defined in relation to the conformance algorithm and corresponding parameters specified in the connection traffic descriptor. In traffic contract, a conformance definition based on the conformance algorithm GCRA is defined. We will explain the details of the GCRA later.

- QoS Parameters

In this subsection, the negotiable QoS parameters are described. They are Maximum Cell Transfer Delay (MaxCTD), Peak-to-Peak Cell Delay Variation (p-to-p CDV), and Cell Loss Ratio (CLR).

1. MaxCTD: The measured Cell Transfer Delay (CTD) is defined as the elapsed time between a cell exit event at the source point and the corresponding cell entry event at the destination point for a particular connection. It is the sum of the total inter-node transmission delay and the total node processing delay between these two points. Fig. 2.1 shows the probability density function of the CTD, and relates it to maxCTD and p-to-p CDV parameters.

   The maxCTD specified for a connection is the $(1 - \alpha)$ quantile of CTD.

2. Peak-to-Peak CDV: The peak-to-peak CDV is the difference between the maxCTD and the fixed CTD as shown in Fig. 2.1. The fixed CTD is experienced by any delivered cell on a connection during the entire connection holding time.

   Two methods are defined to measure the CDV:

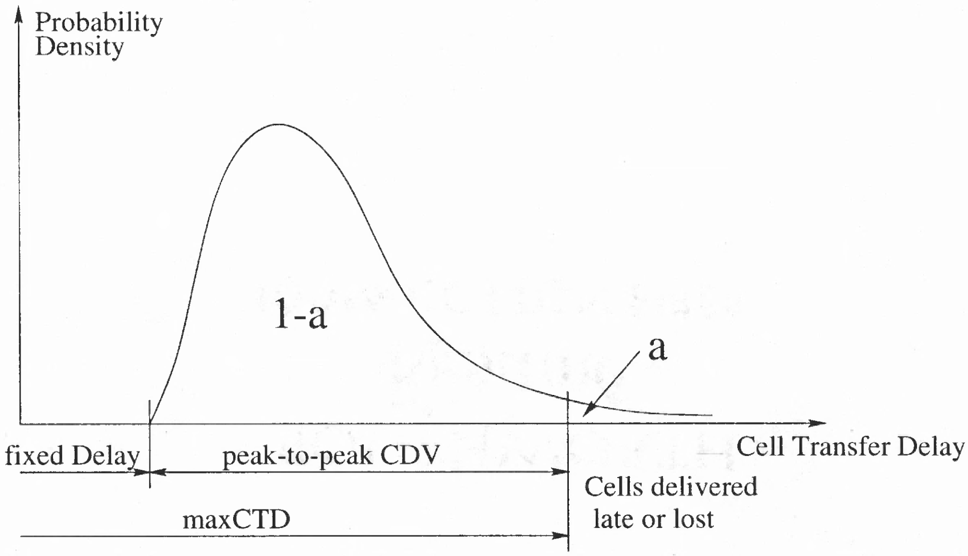


Figure 2.1 Cell Transfer Delay Probability Density Model [34]

– One-point CDV: The one-point CDV for k (y_k) at a measurement point is defined as the difference between the cell's reference time (c_k) and actual arrival time (a_k)at the same measurement point:

$$y_k = c_k - a_k \tag{2.1}$$

. The reference time is defined as:

$$c_0 = a_0 \tag{2.2}$$

$$c_k + 1 = \begin{cases} a_k + T & \text{if } a_k > c_k \\ c_k + T & \text{otherwise} \end{cases} \tag{2.3}$$

Positive one-point CDV means cell clumping phenomenon observed; and the negative one corresponds to gaps in the cell stream. The reference arrival time defined above eliminates the effect of gaps and provides a measurement of cell clumping.

– Two-point CDV: The two-point CDV is the variability in the pattern of cell arrival events observed at the output of a measurement point with reference to the pattern of the corresponding events observed at the input to another measurement point. The two-point CDV for cell k (v_k) between two measurement points is the difference between the absolute CTD of cell k (x_k) between these two points and a defined reference CTD ($d_{1,2}$) between the same points: $v_k = x_k - d_{1,2}$. The absolute CTD (x_k) of cell k between two measurement points is the same as CTD. The reference CTD ($d_{1,2}$) between these two points is the absolute CTD experienced by a reference cell between the two measurement points.

3. Cell Loss Ratio: The Cell Loss Ratio is defined for a connection as:

$$CLR = \frac{Lost\ Cells}{Total\ Transmitted\ Cells} \tag{2.4}$$

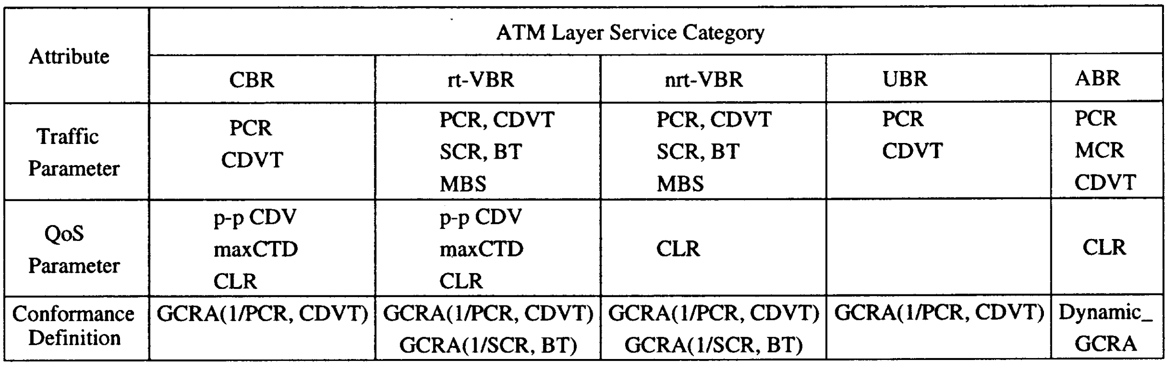
The Cell Loss Ratio parameter is the value of CLR that the network agrees to offer as an objective over the lifetime of the connection.

2.3 ATM Service Classes

ATM Forum's Traffic Management Version 4.0 specifies four main service classes based on the traffic characteristics and QoS requirements. They are Constant Bit Rate (CBR) service, Variable Bit Rate (VBR) service which also includes Real-Time (RT) and Non-Real-Time (NRT) traffic, Unspecified Bit Rate (UBR) service, and Available Bit Rate (ABR) service [34].

Fig. 2.2 shows the list of ATM traffic parameters for each service category.

CBR is a service category for connections that request a static amount of bandwidth that is continuously available during the connection lifetime [34]. The cell intervals in the CBR are almost constant. CBR offers consistent delay predictability of circuit-switched services or leased line services with guaranteed bandwidth allocation. CBR traffic (e.g., voice traffic supported by the conventional telephone network) requires the Peak Cell Rate bandwidth allocation to an ATM



Attribute	ATM Layer Service Category				
	CBR	rt-VBR	nrt-VBR	UBR	ABR
Traffic Parameter	PCR CDVT	PCR, CDVT SCR, BT MBS	PCR, CDVT SCR, BT MBS	PCR CDVT	PCR MCR CDVT
QoS Parameter	p-p CDV maxCTD CLR	p-p CDV maxCTD CLR	CLR		CLR

Conformance Definition	GCRA(1/PCR, CDVT)	GCRA(1/PCR, CDVT)
		GCRA(1/SCR, BT)
			GCRA(1/PCR, CDVT)
			GCRA(1/SCR, BT)
				GCRA(1/PCR, CDVT)	Dynamic_
					GCRA

Figure 2.2 ATM Service Category Attributes [34, 44]

connection [90]. CBR service is intended to support real-time applications which are sensitive to the delay and delay variation [47, 48]. In the CBR capability, the source may emit idle cells at, or below the PCR. It also may delete those cells which are delayed beyond the specified maxCTD [34].

VBR is a service category for the variable-bit-rate-type traffic in which the cell interval is not constant. VBR traffic is described by PCR, SCR, MBS. It does not always require the allocation of the PCR bandwidth. The bandwidth allocation of VBR traffic is based on its PCR and SCR, generally between these two value [90]. Therefore, VBR service is good for the bandwidth utilization than CBR service, in which it is always allocated the PCR bandwidth. VBR includes two types of traffic. They are:

- Real-Time VBR(rt-VBR) service: it is intended for real-time applications which require constrained delay and delay variation. Sources are expected to transmit at a rate which varies with time. It means that the source can be described as "bursty" [34]. Its QoS requirements include the maxCTD. It can support statistical multiplexing of real-time sources, and get a relatively high bandwidth utilization. It was studied in [3, 57, 59, 88].

- Non-Real-Time VBR(nrt-VBR) service: it is intended for non-real-time applications which have bursty traffic characteristics. Cells experience a low cell loss ratio when they are conforming to the traffic contract. No delay bounds are specified in this service category [34].

UBR is a service category for non-real-time application. In UBR, the CLR and CTD of each connection are not guaranteed. Although it is also described by PCR, the value of PCR is just informational. The network may not allocate bandwidth based on its PCR. The UBR service is served whenever the network resources are available. It is like a best-effort traffic [1, 49, 50, 65]. The UBR traffic will be dropped first when the network is congested [34, 90].

ABR is a service that its ATM layer transfer characteristics may change based on the available network resources during the connection lifetime. A flow control mechanism must be used to control the source rate in response to changing ATM layer transfer characteristics [21, 50, 93, 100]. In order to get the information of available network resources, several types of feedback are supported in ABR capability. The ABR services doesn't guarantee the delay and delay variation, so it can not be for real-time applications. The ABR is described by PCR and MCR, which designate the maximum required bandwidth and a minimum usable bandwidth, respectively. The available bandwidth of the network may vary, but shall not become less than MCR [34, 90].

2.4 Traffic Shaping

In ATM networks, if there is any cell non-conforming to the traffic contract established at the instance of acceptance of the connection, network operators can drop or tag the non-conforming cells. They also can use traffic shaper to shape the stream, that is, delay the non-conforming cell till its conforming Departure Time(DT). There are many algorithms that can be used to calculate the conforming DT, such as, jumping window [60, 72, 92], moving window [44], exponentially weighted moving average [28, 72] etc.. However, the most widely accepted traffic shaping and scheduling algorithm is Generic Cell Rate Algorithm (GCRA), also known as Leaky Bucket Algorithm.

2.4.1 Generic Cell Rate Algorithm

The GCRA is used to define conformance with respect to the traffic contract. It is well known as Leaky Bucket Algorithm [13, 15]. The GCRA is a virtual scheduling algorithm or a continuous-state leaky-bucket algorithm. The GCRA is defined with two parameters: the Increment (I), and the Limit (L), and denoted as GCRA(I, L). The basic process of this algorithm is described by ATM Forum [34] as in Fig. 2.3.

The explanation for the flowchart follows. The virtual scheduling algorithm updates the Theoretical Arrival Time (TAT), which is the nominal arrival time of the cell assuming equally spaced cells when the source is active. If the actual arrival time of a cell is not too early compared to the TAT, in particular the actual arrival time is not earlier than $TAT - L$, then the cell is conforming. Otherwise, the cell is not conforming. The continuous-state leaky-bucket algorithm can be viewed as finite-capacity bucket whose real-valued content drains out at a continuous rate of 1 unit of content per time unit and whose content is increased by the increment I for each conforming cell. If at a cell arrival the content of the bucket is less than or equal to the limit value, L, then the cell is conforming. Otherwise, the cell is non-conforming. The capacity of the bucket is $L + I$ [34, 44]. The depiction of the GCRA(I,L) is shown in the Fig. 2.4.

According to the individual connection's traffic contract, traffic shaper assigns a conformance time to each incoming cell using the GCRA. When a cell arrives, the GCRA updates the leaky bucket level X' according to the new arrival time $RAT(k)$

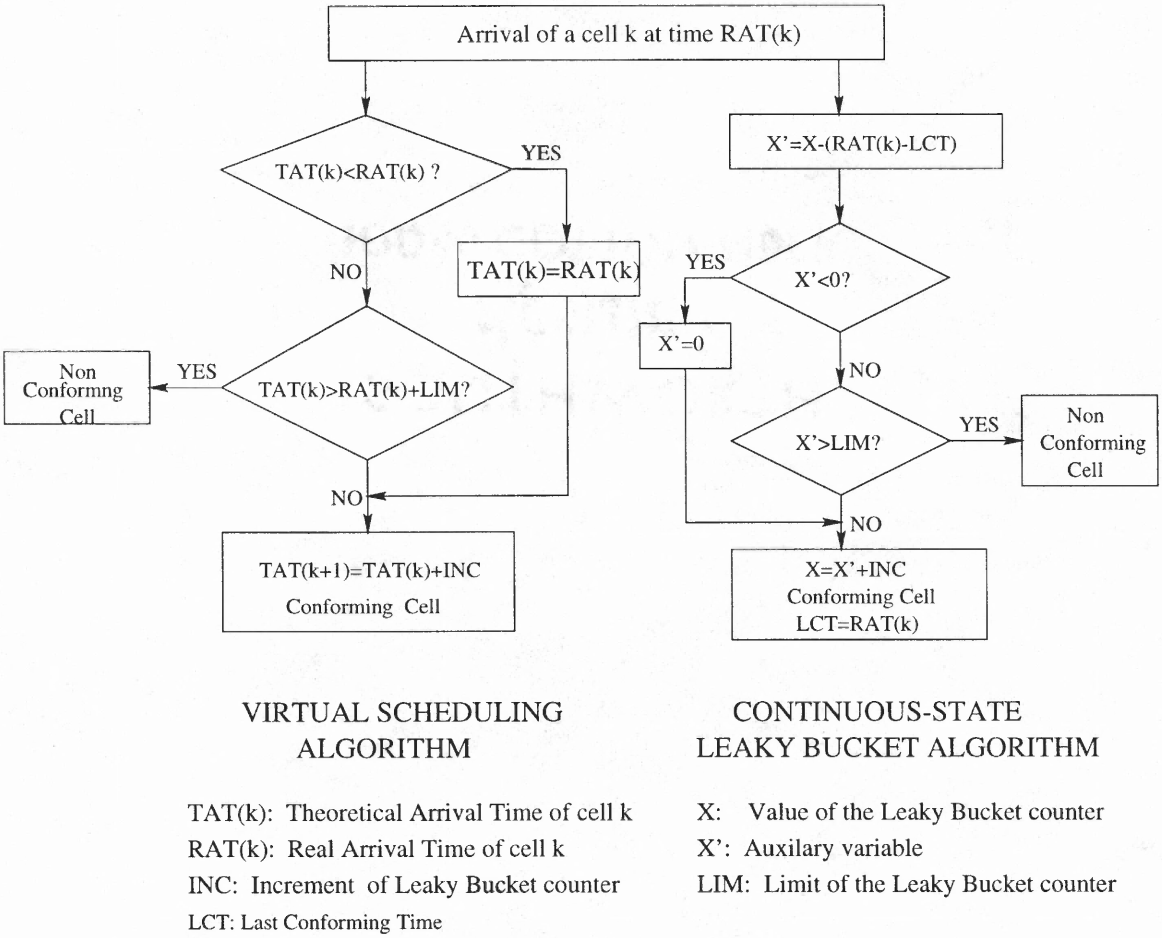

Arrival of a cell k at time RAT(k)
TAT(k)<RAT(k) ?
YES
TAT(k)=RAT(k)
NO
Non Conformng Cell
YES
TAT(k)>RAT(k)+LIM?
NO
TAT(k+1)=TAT(k)+INC
Conforming Cell
X'=X-(RAT(k)-LCT)
YES
X'<0?
X'=0
NO
X'>LIM?
YES
Non Conforming Cell
NO
X=X'+INC
Conforming Cell
LCT=RAT(k)
VIRTUAL SCHEDULING ALGORITHM
CONTINUOUS-STATE LEAKY BUCKET ALGORITHM
TAT(k): Theoretical Arrival Time of cell k
RAT(k): Real Arrival Time of cell k
INC: Increment of Leaky Bucket counter
LCT: Last Conforming Time
X: Value of the Leaky Bucket counter
X': Auxilary variable
LIM: Limit of the Leaky Bucket counter

Figure 2.3 Generic Cell Rate Algorithm [34]

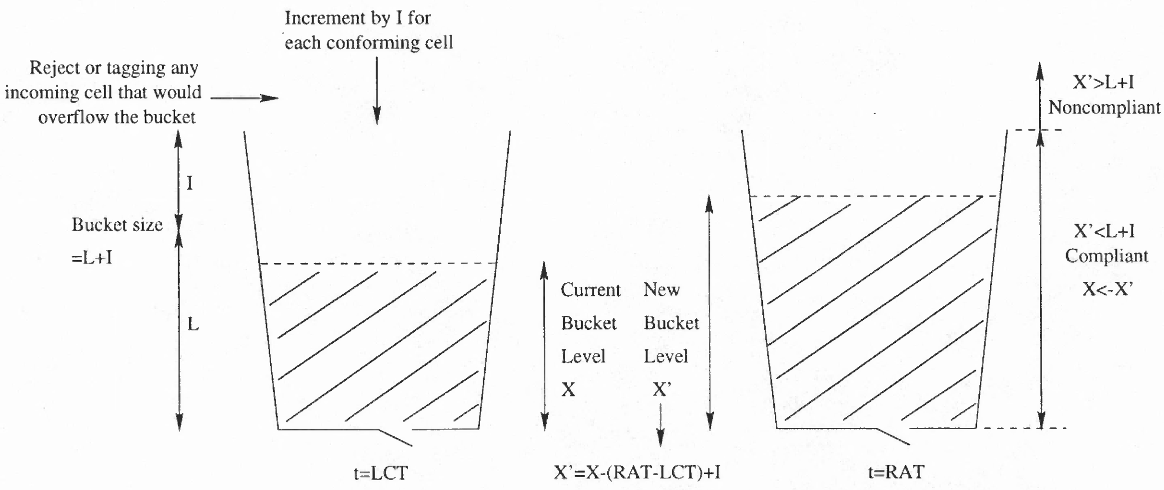


Figure 2.4 Depiction of GCRA(I, L) [81]

and the Last calculated Compliance Time LCT,

$$X' = X - (RAT(k) - LCT). \tag{2.5}$$

If the updated leaky bucket level X' exceeds the limit of bucket (L), then this is a non-conforming cell, and should be delayed at least $X' - L$. Otherwise, this is a conforming cell, and should be assigned a conformance time equal to $RAT(k)$ [69].

Generally, the dual leaky bucket algorithm is employed to control the bursty data service [43]. There are four traffic parameters associated with dual leaky bucket algorithm, PCR, CDVT, SCR, BT. The dual leaky bucket algorithm is denoted by $GCRA(T_p, \tau_p, T_s, \tau_s)$, in which, T_p is the Peak Emission Interval, T_s is the Sustained Emission Interval, $T_p = \frac{1}{PCR}, \tau_p = CDVT, T_s = \frac{1}{SCR}, \tau_s = BT$. One of the leaky buckets is defined by T_p and τ_p, which is used to eliminate the sustained high rate but small burst size phenomenon, another is defined by T_s and τ_s, which is used to control the sustained medium rate but with very large burst size phenomenon. When there is a cluster of cells coming at a high speed, even the length of the cluster is small, it may make the bucket associated with the PCR full. Then, the cells are delayed so that the rate is reduced, and congestion may be avoided. In another case, although the rate of the cells are moderate, just little higher than the SCR, however, if the length of burst is very large, it also may lead to the leaky bucket associated with the SCR full. Then, this SCR bucket can control such a kind of bursts. Therefore, the dual leaky bucket algorithm can control the bursty data much more efficient than single leaky bucket algorithm. The algorithm of each leaky bucket is as same as Fig. 2.3 describes. The leaky bucket levels are denoted by X_s, X_p, and updated leaky bucket levels are denoted by X'_s, X'_p respectively.

For an incoming cell, if it does not conform to anyone of the leaky buckets, it is judged as non-conforming cell, and should be assigned a conforming departure time

$$DT = RT + max(X'_s - \tau_s, X'_p - \tau_p), \tag{2.6}$$

in which, RT is the Real Time or current time.

2.4.2 Event-Driven Traffic Shaping Algorithms

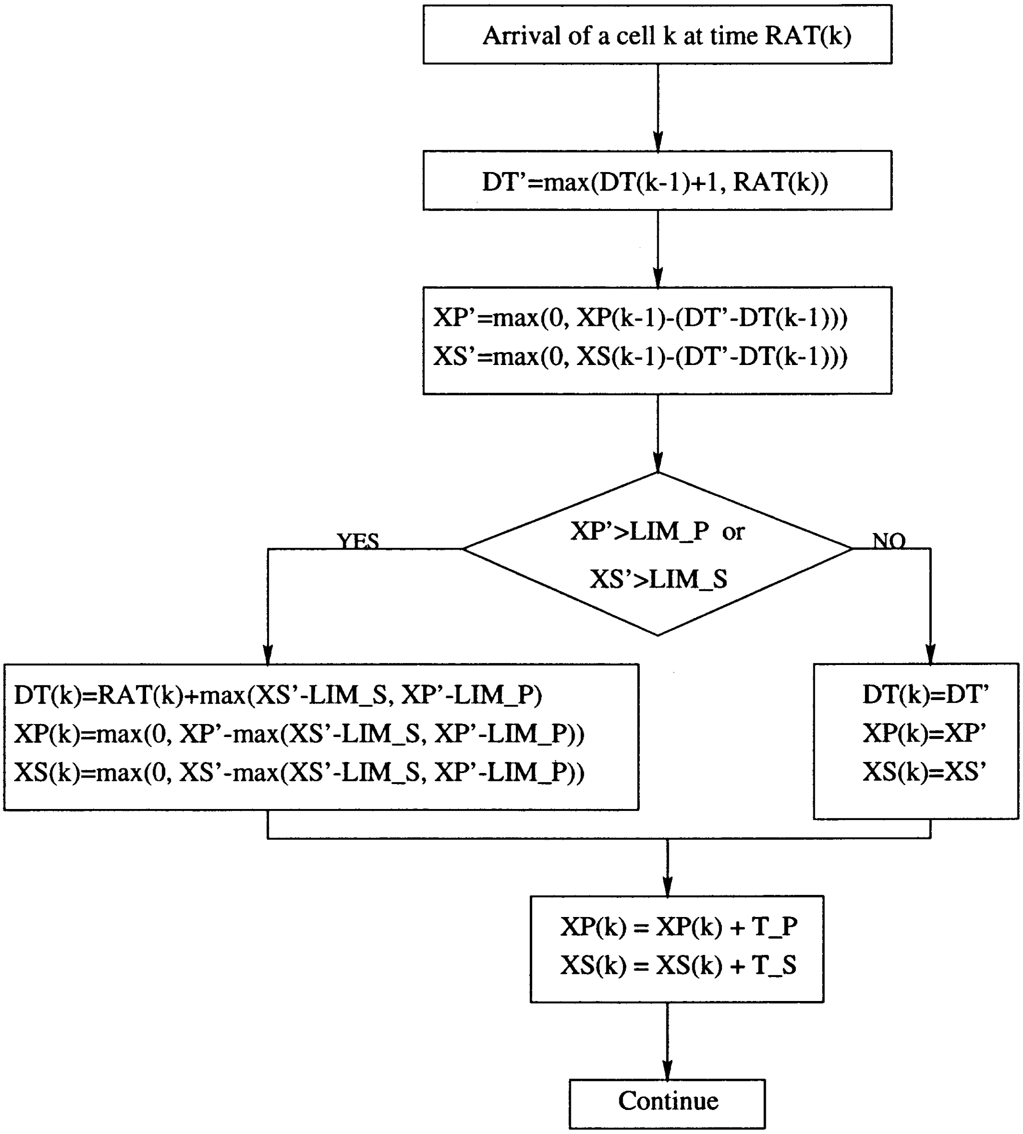
Based on the GCRA, the traffic shaper assigns conforming DT to each incoming cell so that it can schedule their transmission. For the assignment of departure time, two event-driven traffic shaping algorithms are proposed: Arrival Event Driven Traffic Shaping algorithm and Departure Event Driven Traffic Shaping algorithm [15, 18, 44].

2.4.2.1 Arrival Event Driven Traffic Shaping Algorithm (AEDTS) Arrival Event Driven Traffic Shaping Algorithm (AEDTS) is the conventional leaky bucket algorithm which is recommended by ATM Forum [34]. In AEDTS, while a new cell is received and appended to its associated Virtual Connection Queue (VCQ), its departure time will be calculated based on the calculated DT of the previous cell in the same VCQ using GCRA. The leaky bucket level will be updated at the same time [18, 44]. The algorithm is shown in Fig. 2.5.

An example of AEDTS is shown in Fig. 2.6.

In this example, there is only one connection. This connection's increment I=4, and the limit L=2, that is T=4, and $\tau = 2$, the capacity of the bucket is $T + \tau = 4 + 2 = 6$. The output of the traffic shaper is a strict conforming traffic flow. No departure cell makes the leaky bucket overflow.

However, for AEDTS, although the output of each connection complies with its own leaky bucket parameters, as multiplexed on the outlink, each shaped cell may violate the shaping parameter because of the contention among eligible cells for transmission [15]. If the previous cell of one VCQ is delayed due to loss of contention, the successive cell of the same VCQ may be assigned an improper DT based on the calculated DT of the previous cell which is earlier than its real departure time. Such


Arrival of a cell k at time RAT(k)
DT'=max(DT(k-1)+1, RAT(k))
XP'=max(0, XP(k-1)-(DT'-DT(k-1)))
XS'=max(0, XS(k-1)-(DT'-DT(k-1)))
XP'>LIM_P or
XS'>LIM_S
YES
NO
DT(k)=RAT(k)+max(XS'-LIM_S, XP'-LIM_P)
XP(k)=max(0, XP'-max(XS'-LIM_S, XP'-LIM_P))
XS(k)=max(0, XS'-max(XS'-LIM_S, XP'-LIM_P))
DT(k)=DT'
XP(k)=XP'
XS(k)=XS'
XP(k) = XP(k) + T_P
XS(k) = XS(k) + T_S
Continue

RAT: Real Arrival Time; DT: Departure Time; DT', XP', XS': Auxiliary Variable
XP: Level of the Leaky Bucket associated with Peak_cell_rate;
LIM_P: Limit of the Leaky Bucket associated with Peak_cell_rate;
XS: Level ofthe Leaky Bucket associated with Sustainable_cell_rate;
LIM_S: Limit of the Leaky Bucket associated with Sustainable_cell_rate;
T_S: Increasement interval of the Leaky Bucket associated with PCR
T_P: Increasement interval of the Leaky Bucket associated with SCR;

Figure 2.5 Arrival Event Driven Traffic Shaping Algorithm

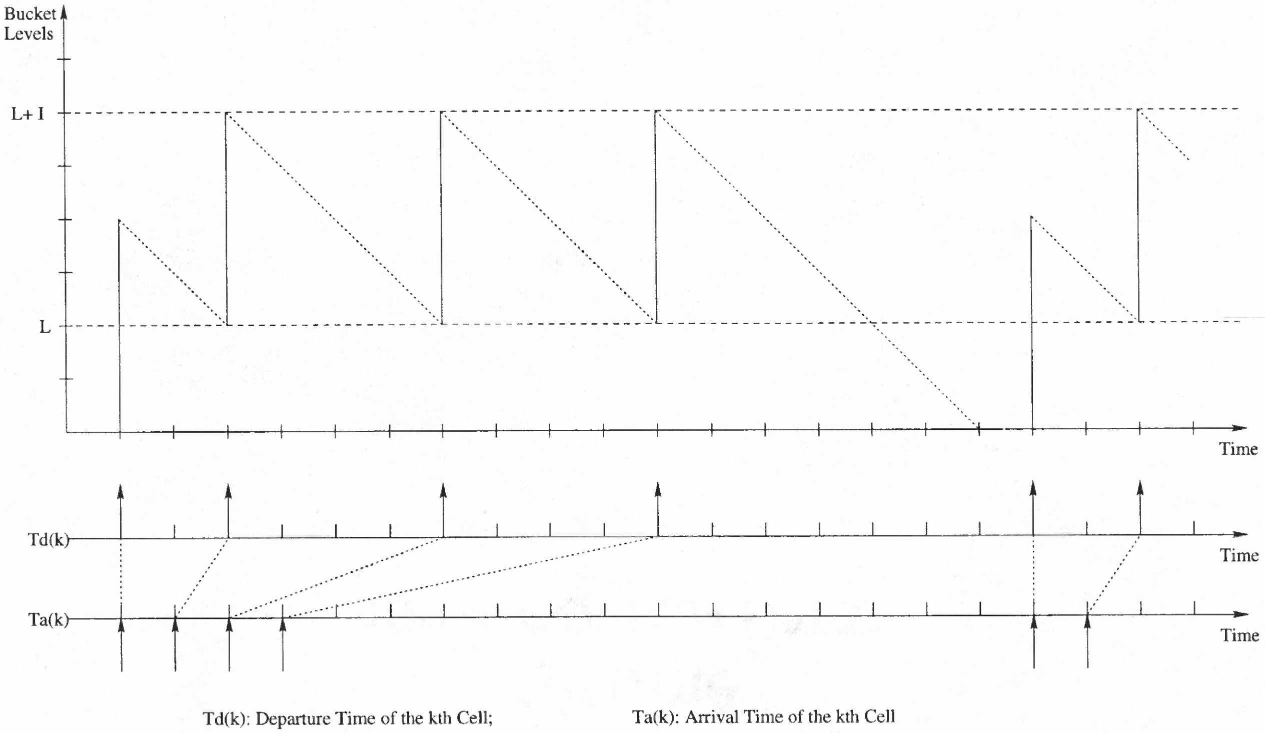

Bucket Levels
L+ I
L
Time
Td(k)
Time
Ta(k)
Time
Td(k): Departure Time of the kth Cell;
Ta(k): Arrival Time of the kth Cell

Figure 2.6 An Example of DT Calculation Diagram for AEDTS (L=2, I=4, GCRA(4, 2))

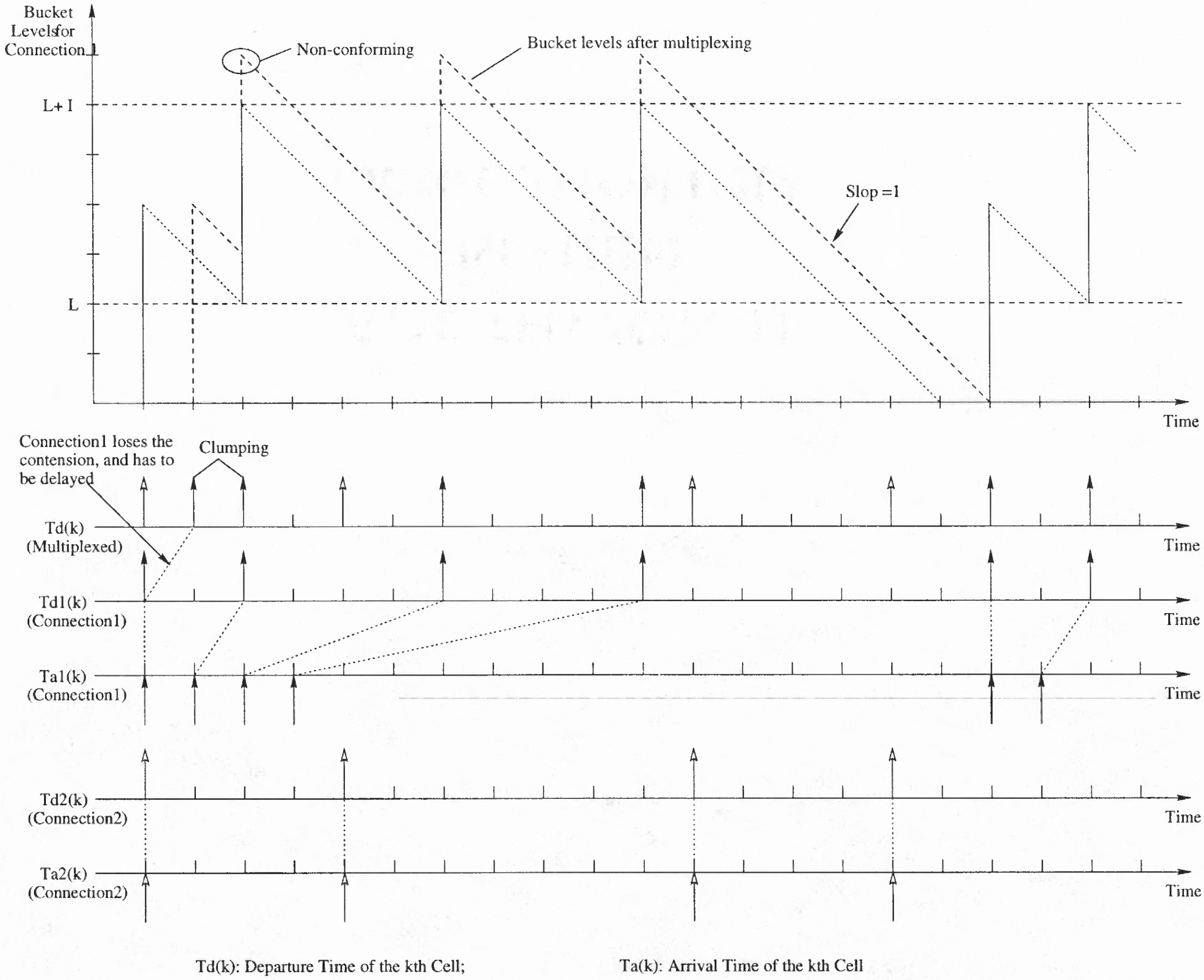

Bucket Levelsfor Connection 1
Non-conforming
Bucket levels after multiplexing
L+ I
Slop =1
L
Time
Connection1 loses the contension, and has to be delayed
Clumping
Td(k) (Multiplexed)
Time
Td1(k) (Connection1)
Time
Ta1(k) (Connection1)
Time
Td2(k) (Connection2)
Time
Ta2(k) (Connection2)
Time
Td(k): Departure Time of the kth Cell;
Ta(k): Arrival Time of the kth Cell

Figure 2.7 Clumping Phenomenon in AEDTS (both connections with L=2, I=4)

improper assignment of DT may clump the successive cell sequence so that it leads to a larger Cell Delay Variation which may violate the traffic contract CDVT and BT. This problem is shown in Fig. 2.7.

In this scenario, after the multiplexing, due to the first cell of connection 1 loses its contention, and has to be delayed one cell slot, the first two cells of connection 1 are clumped together, and violate the traffic contract, overflow the bucket. It shows that the AEDTS may not guarantee the multiplexed connections strict compliant to their individual traffic contract any more.

2.4.2.2 Departure Event Driven Traffic Shaping Algorithm (DEDTS)

In order to strictly limit the CDVT and BT at the output of the shaping multiplexer [15, 44], Departure Event Driven Traffic Shaping Algorithm (DEDTS) is proposed. Compared to the traditional GCRA which is AEDTS, in DEDTS, bucket level update and DT calculation are performed only when the previous cell of the same connection departs from the multiplexer. The algorithm is shown in Fig. 2.8. An example of DT calculation of DEDTS is shown in Fig. 2.9.

In this example, there is only one connection. It is same as in AEDTS in this scenario. However, when multiple connections are multiplexed together, the DEDTS resolves the clumping problem which is shown in Fig. 2.10.

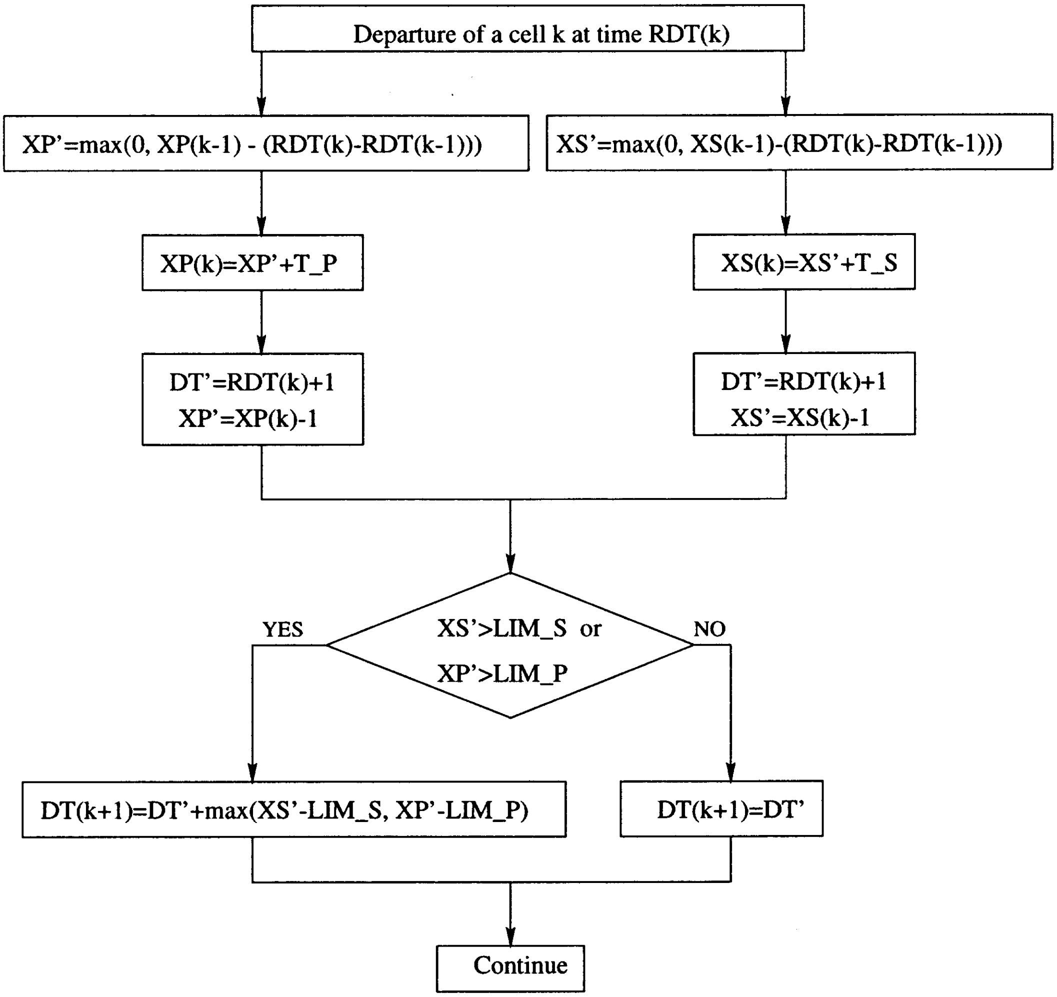
Because the DT is calculated based on the real DT of the previous cell of the same connection, DEDTS can guarantee strict conforming to CDVT value of each connection. Meanwhile, in each time slot, only one cell is transmitted, so only one new conformance time DT is calculated, which also eliminates the bottleneck due to multiple DT calculation for more than one cell arriving at the same cell slot in AEDTS.

2.5 QoS in Internet

With the Internet evolves from a best effort network to a network supporting QoS, the IETF has proposed many service models and mechanisms to meet the demand for QoS. Integrated Services (IntServ) framework and Differentiated Services (DiffServ) framework are two of the most important proposals.

2.5.1 Integrated Services and RSVP

RSVP is a signaling protocol that can be used by hosts to request resource reservations through a network, and set up paths [4, 91, 101]. The IntServ model is a framework for providing end-to-end services in the context of RSVP signaling. It is


Departure of a cell k at time RDT(k)
XP'=max(0, XP(k-1) - (RDT(k)-RDT(k-1)))
XS'=max(0, XS(k-1)-(RDT(k)-RDT(k-1)))
XP(k)=XP'+T_P
XS(k)=XS'+T_S
DT'=RDT(k)+1
XP'=XP(k)-1
DT'=RDT(k)+1
XS'=XS(k)-1
YES
XS'>LIM_S or
XP'>LIM_P
NO
DT(k+1)=DT'+max(XS'-LIM_S, XP'-LIM_P)
DT(k+1)=DT'
Continue

RDT: Real Departure Time;
DT: Assigned Departure Time;
T_P, T_S: Increasement of Leaky Buckets;
DT', XP', XS': Auxiliary Variable

XP: Level of the Leaky Bucket associated with PCR;
LIM_P: Limit of the Leaky Bucket associated with PCR;
XS: Level of the Leaky Bucket associated with SCR;
LIM_S: Limit of the Leaky Bucket associated with SCR;

Figure 2.8 Departure Event Driven Traffic Shaping Algorithm

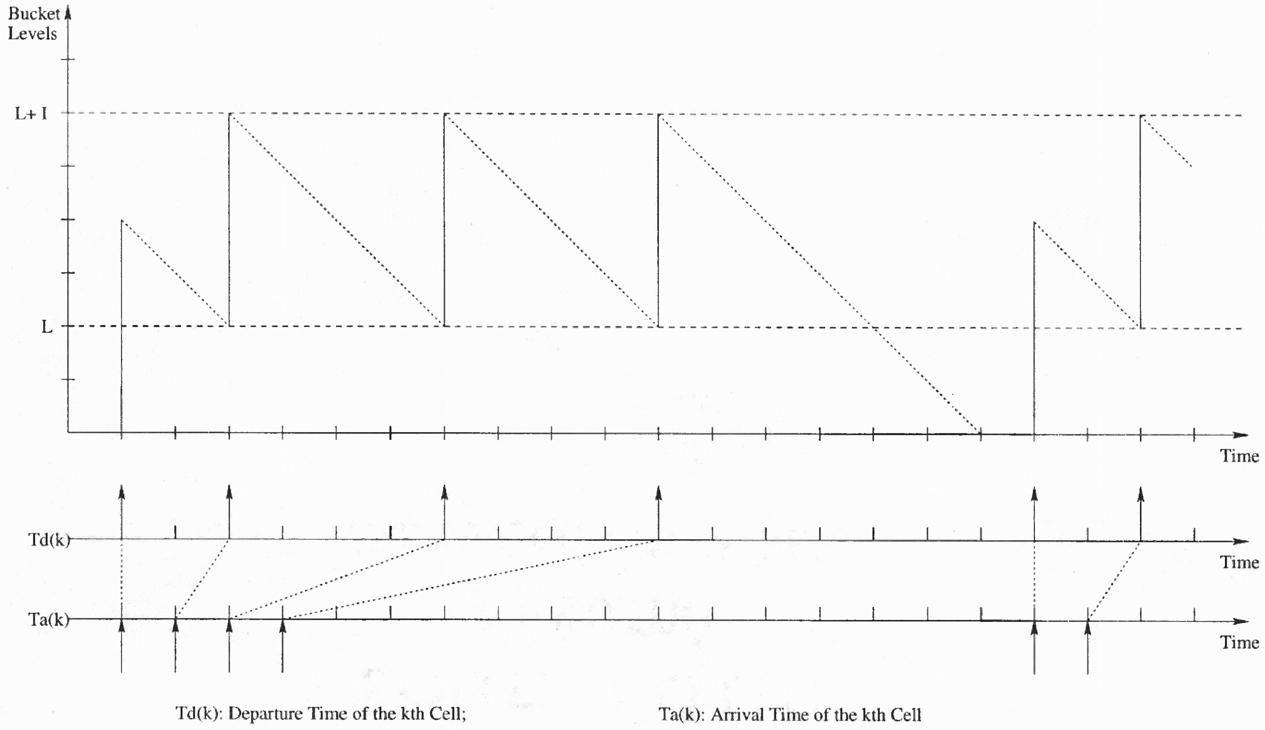

Bucket Levels
L+ I
L
Time
Td(k)
Time
Ta(k)
Time
Td(k): Departure Time of the kth Cell;
Ta(k): Arrival Time of the kth Cell

Figure 2.9 An Example of DT Calculation Diagram of DEDTS (L=2, I=4)

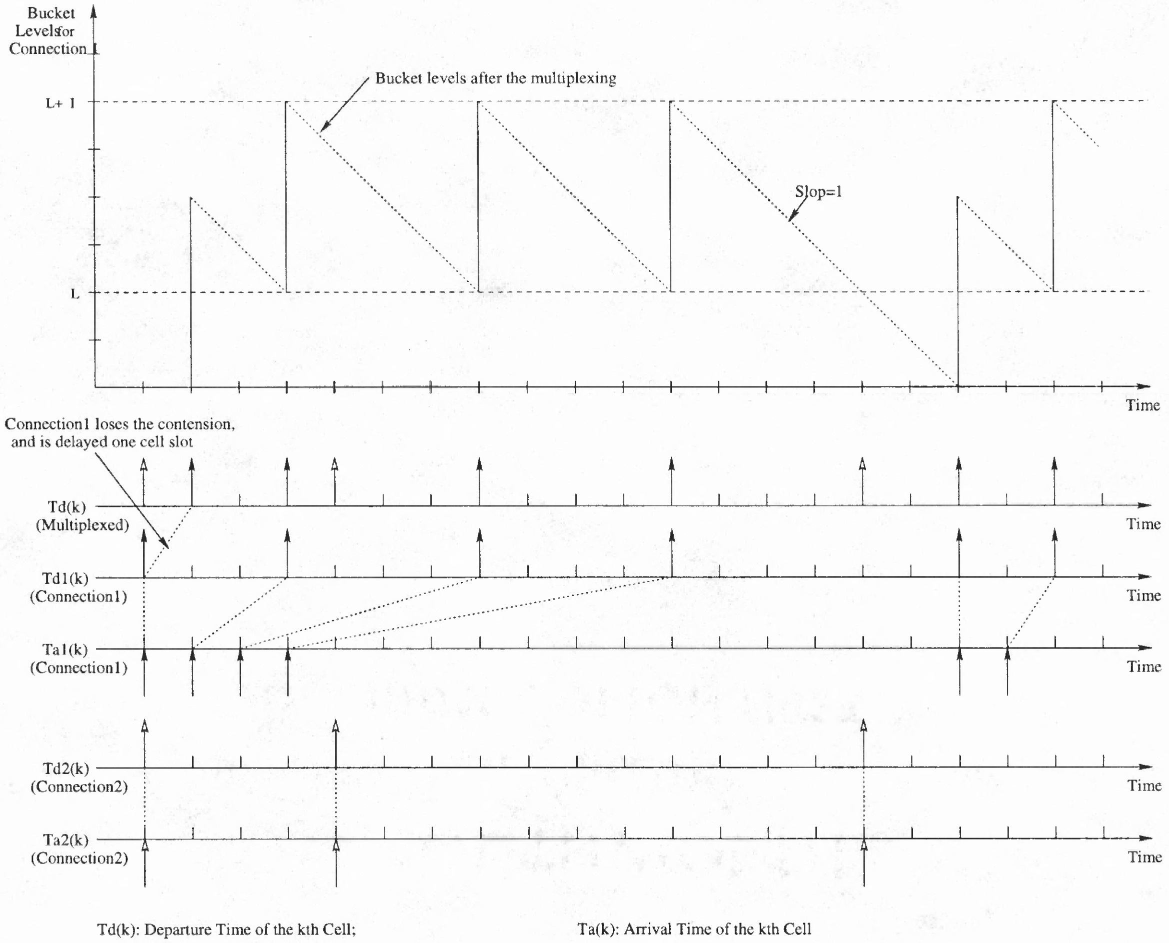

Bucket
Levels for
Connection 1
Bucket levels after the multiplexing
L+ I
Slop=1
L
Time
Connection1 loses the contension,
and is delayed one cell slot
Td(k)
(Multiplexed)
Time
Td1(k)
(Connection1)
Time
Ta1(k)
(Connection1)
Time
Td2(k)
(Connection2)
Time
Ta2(k)
(Connection2)
Time
Td(k): Departure Time of the kth Cell;
Ta(k): Arrival Time of the kth Cell

Figure 2.10 An Example of DT Calculation of DEDTS in Multiplexing Scenario (both connections with L=2, I=4)

characterized by resource reservation. IntServ assumes that network devices support traffic handling mechanisms, which guarantee service to each traffic flow in strict isolation from other traffic flows. In the integrated services model [9], there are three service classes. They are:

- Guaranteed Service: it is for applications requiring fixed delay bound [76, 91];
- Controlled-load Service: it is for applications requiring reliable and enhanced best-effort service [89, 91];
- Best-effort Service: it is for the traditional Internet applications.

The philosophy of this model is that "There is an inescapable requirement for routers to be able to reserve resources in order to provide special QoS for specific user packet streams, or flows. This in turn requires flow-specific state in the routers" [9, 91]. RSVP was invented to reserve resources for applications. The RSVP signaling process is illustrated in Fig. 2.11.

The whole process is described as the following:

1. the sender generates RSVP message known as the PATH message. The PATH message carries the information for the traffic description. It includes the identity of the sender, the sending application, the traffic profile, and the classification criteria by which the traffic can be recognized [4].
2. The sender sends this PATH message to the receiver specifying the characteristics of the traffic. Every intermediate network device, such as a router, along the path installs the PATH state, and forwards the PATH message to the next hop determined by the routing protocol.
3. When the receiver gets this PATH message, it responds with a RESV message to request resources for the flow. Every intermediate router along the path can

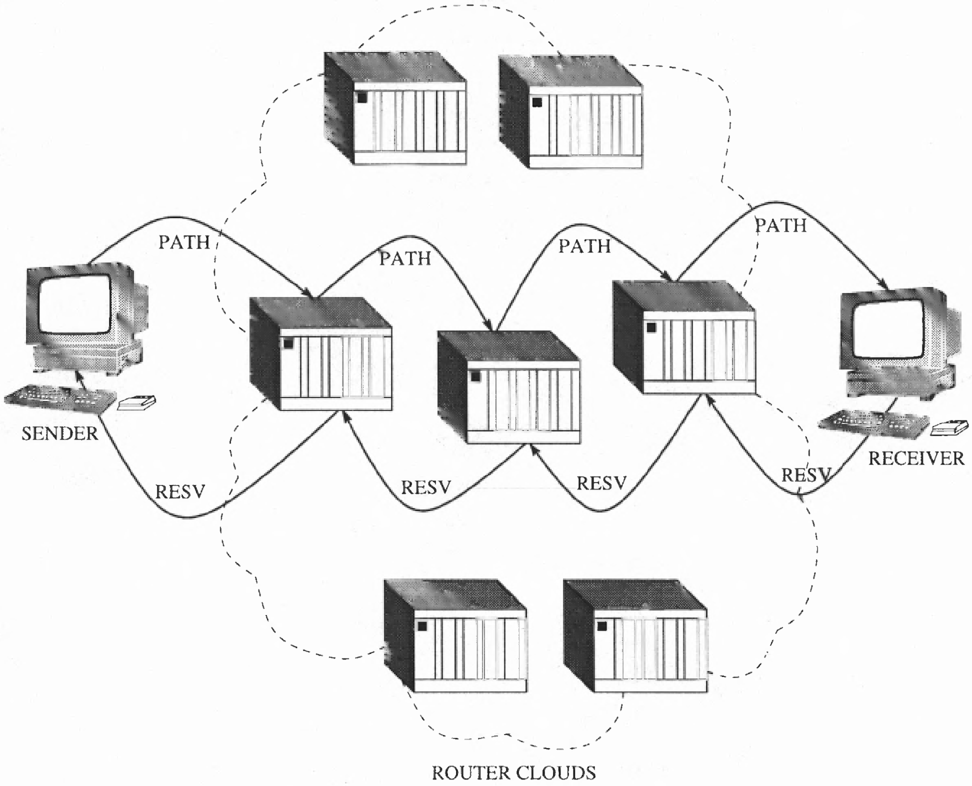


Figure 2.11 RSVP Signaling Process

apply admission control to determine whether the request of the RESV message is accepted or not. If the request is accepted, link bandwidth and buffer space are allocated for the flow, and related flow state information will be installed in the network device. When a device rejects a request, error messages are sent along the data path, notifying other RSVP-aware devices of the admission control failure [4, 91].

Recently, the RSVP has been modified and extended in several ways to reserve resources for aggregation of flows, to set up explicit routes (ERs) with QoS requirements, and to do some other signaling tasks [32, 38, 63].

IETF proposed a reference implementation framework to realize the IntServ/RSVP model. This framework consists of four components: the packet scheduler, the admission control routine, the classifier, and the reservation setup protocol [9]. The implementation reference model is shown in Fig. 2.12.

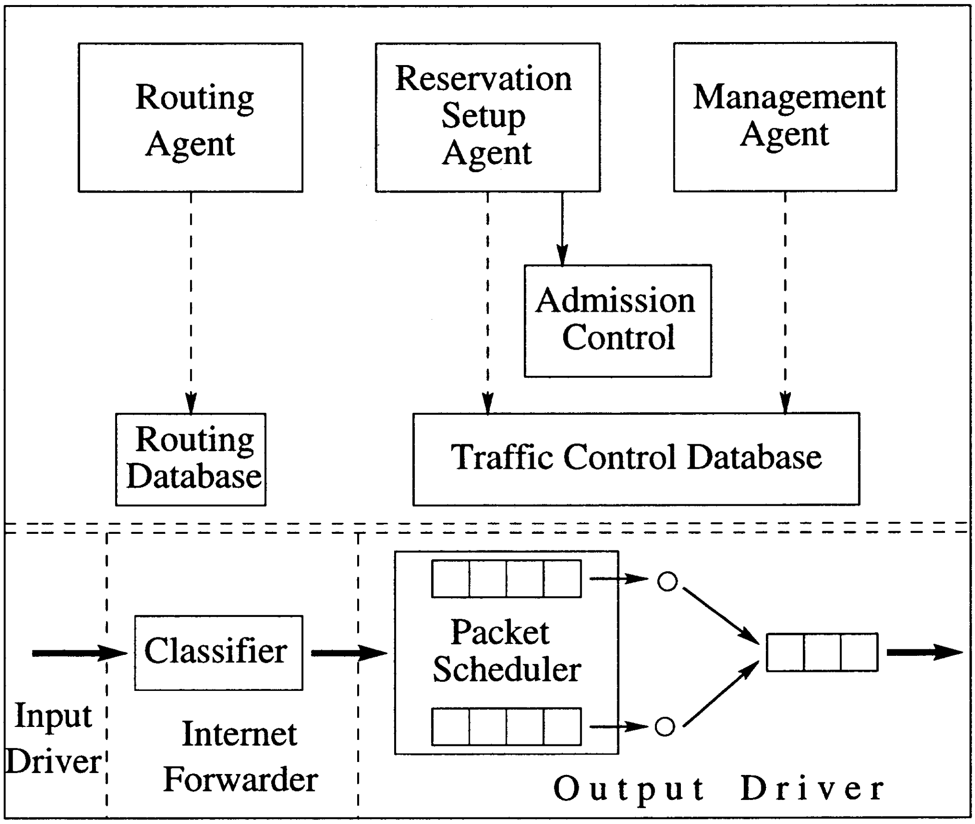


Figure 2.12 The Implementation Reference Model of IntServ/RSVP [9]

The packet scheduler manages the forwarding of different packet streams using a set of queues and queue management policy, and runs some scheduling algorithm. The classifier classifies or maps each incoming packet into different classes, and all packets in the same class get the same treatment from the packet scheduler. The classifying or mapping may be based upon the contents of the existing packet header(s) and/or some additional classification number added to each packet. The admission control determines whether there are enough resources for the new flow without impacting the existing connections in the network. Based on this decision, the network device chooses to accept or reject the new flow. The fourth component is the RSVP. It sets up a path for the new flow, as well as creates and maintains flow-specific state in the endpoint hosts and in network devices along the path of the flow [9].

The IntServ network model makes a fundamental change to the concept of the Internet architecture. It requires that all flow-related information should be in the end systems. This scheme can guarantee the QoS for each flow [25, 91]. However, it also leads to some problems:

- The amount of state information increases proportionally to the number of flows. In the current Internet, there can be thousands, ten thousands, even hundred thousands of flows simultaneously. This architecture is definitely unscalable [91].
- The implementation complexity of the network device is very high. Each device must have the packet scheduler, classifier, admission control, and RSVP. It also leads to the scalability problem [4, 91].

2.5.2 Differentiated Services

2.5.2.1 Motivation for DiffServ Because of the difficulty in implementing and deploying the IntServ/RSVP architecture, differentiated services (DiffServ) architecture is proposed by the IETF to resolve the scalability problem in the IntServ/RSVP model.

DiffServ is a traffic handling mechanism [4]. It defines a field in packets' IP headers, called the diffserv codepoint (DSCP). It is six-bit wide in IPv6, and three-bit wide in IPv4 which is also called type of service (TOS) in IPv4 header. In addition, it defines a base set of packet forwarding treatments, called the per-hop behavior, or PHBs [66]. By marking the DSCP of packets differently, the network devices can classify packets and apply specific queueing behavior based on the results of the classification. Traffic from multiple flows having similar QoS requirements is marked with the same DSCP, thus aggregating the flows to a common queue or scheduling behavior. The distinguishing feature of DiffServ is its scalability [4]. It is because that the aggregate traffic handling mechanisms require significantly less state information and processing capability in network nodes than their per-flow counterparts.

2.5.2.2 Configuration of a DiffServ Network In the DiffServ model proposed by the IETF, the DiffServ network is configured to provide many services to its customers. These services are specified by the *service level agreement* (SLA). The SLA is defined at the instance that the customers submit traffic to the DiffServ network's ingress router. An SLA may specify the traffic profile, service class that the submitted traffic belongs to, QoS requirements, etc. The administrator of the DiffServ network will use whatever provisioning and configuration tools are available to meet the SLA for those conforming traffic [4]. An SLA can be static or dynamic. Static SLAs are negotiated on a regular time interval, which is a generally relative long term. Customers with dynamic SLAs must use a signaling protocol (e.g. RSVP) to request services on demand [91].

In a DiffServ network, routers at the ingress point would be configured to classify traffic based on the IP addresses, ports in packet headers and its SLA. The routers would be configured to mark the DS field of packets with one of DSCPs based on the classification results. The internal routers of the DiffServ networks would be configured to classify packets based on DSCP and direct them to a corresponding queue. Queues would be configure to satisfy certain service characteristics specified by the SLAs [4].

In order to provide the QoS guarantee to the conforming traffic, the traffic conditioning must be executed. The traffic conditioning is a set of control functions that can be applied to a behavior aggregate, application flow, or other subset of traffic. They may include metering, policing, shaping, and packet marking as shown in Fig. 6.1 [5, 66]. The classification, policing, scheduling, and shaping rules used at the ingress routers are derived from the SLAs [91]. The traffic conditioning is typically deployed in the ingress routers. Generally, the packets are classified, policed, and possibly shaped and scheduled in DiffServ boundary nodes.

2.5.2.3 Service Classes in DiffServ Networks In a DiffServ network, many services can be provided. They are include:

- Premium Service: It provides low-delay and low-jitter service for customers that generate fixed peak bit rate traffic. Premium service is suitable for those real-time applications, such as Internet telephony, videoconferencing, etc. The SLA for the premium services should specify a desired peak bit-rate for a specific flow or an aggregation of flows. The Internet service provider (ISP) guarantees that the contracted bandwidth will be available when the conforming traffic is sent.

- Assured Service: It is for applications requiring better reliability than best-effort service even in times of network congestion. The SLAs of assured services will specify the amount of bandwidth allocated for the customers, and customers themselves determine how they share that amount of bandwidth.

- Olympic Service: It provides three tiers of services: gold, silver, and bronze, with decreasing quality [40, 67, 91].

2.5.2.4 Characteristics of the DiffServ Model DiffServ model has two most distinguishing features different from the IntServ model. They are:

- One is that there are only a small limited number of service classes indicated by the DS field, and the amount of state information is proportional to the number of classes rather than the number flows, so the management complexity for the state information and the device processing power are reduced greatly.

- Another one is that sophisticated classification, marking, policing, and shaping operation are only needed at the boundary of the networks. Those core network devices only have to deal with behavior aggregate (BA) classification. Consequently, it is easier and more cost efficient to implement and deploy DiffServ networks, and resolve the scalability problem in the IntServ model [4, 91].

2.5.2.5 Access Control at the Edge of the DiffServ Networks In DiffServ networks, all traffic control mechanisms are class-based. However, access networks, such as local area networks, may deal with traffic as individual flows, so the convergence at the edge of the DiffServ network is an important issue. In this dissertation, the traffic shaping mechanism is extended to the DiffServ network environment to implement the fair convergence at the edge of the DiffServ networks, and an implementation scheme for the access control is proposed.

CHAPTER 3

THE HIERARCHICAL TIMING QUEUES TRAFFIC SHAPER

3.1 Motivation for the Hierarchical Timing Queues Traffic Shaper

There are many architectures proposed to implement shaping function. These architectures are classified as either Direct Exact Sorting [15, 18, 44], or Rate Based Grouping architectures in this thesis [73]. In this section, the existing traffic shaper architectures are described, and their problems are pointed out.

3.1.1 Existing Traffic Shaper Architectures

3.1.1.1 Direct Exact Sorting Architecture Direct Exact Sorting architecture is shown in Fig. 3.1. This architecture employs Per-Virtual-Connection-Queue (Per-VCQ) structure at the input port. Following the Per-VCQ, there is a sorting unit. It is a set of timing queues which is formed based on the incoming cell's conforming Departure Time(DT). At the output port, a Departure Queue (DQ) is used to store cells with the conformance time due or overdue.

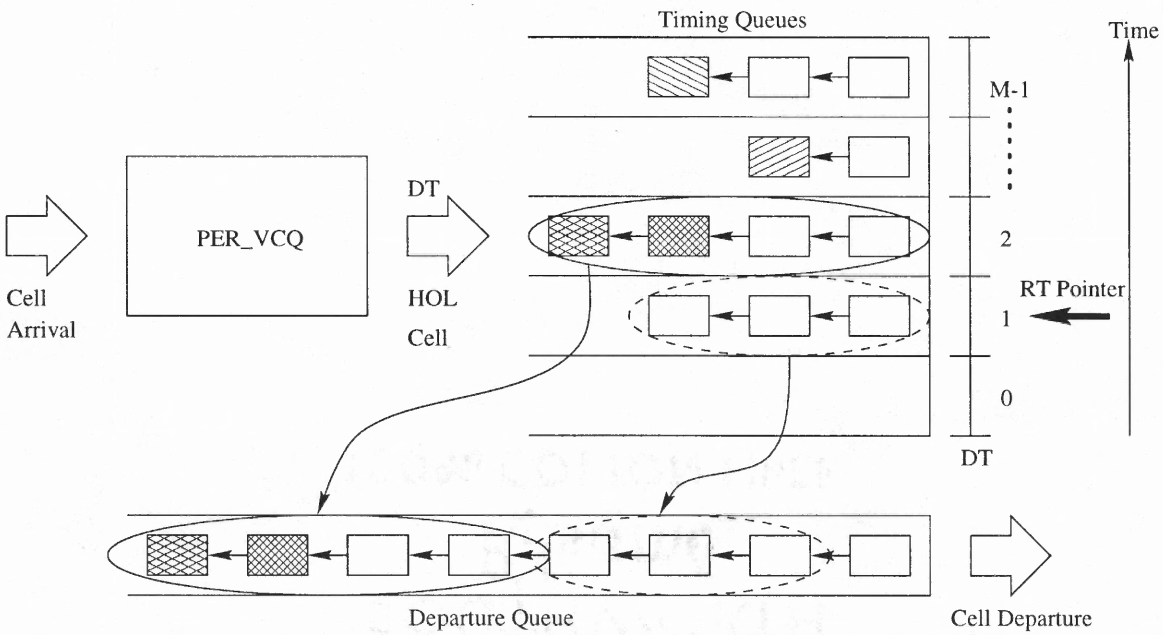


Figure 3.1 Direct Exact Sorting Architecture

In each cell slot, if the DQ is not empty, its Head of Line (HoL) cell with connection ID i will be served. Then, according to the traffic contract of connection i, a new DT of this connection will be calculated by its associated dual leaky buckets. Based on the new DT, the next HoL cell of connection i will be appended to one of timing queues. Once the real time pointer RT points to a timing queue, all cells in the queue become due. Hence, they are appended to the DQ. In the Direct Exact Sorting architecture, all cells with the same DT are appended to the same timing queue, so the DT of a cell becomes the sequence number indicating which timing queue the cell should be appended to. This is the key point of this architecture.

The timing queue scheme can reduce the implementation complexity of the exact sorting of time stamps. Especially, it avoids comparison and insertion operations which are time-consuming. Due to DEDTS, it only needs to calculate one time stamp value per cell slot. However, for hardware implementation, the number of the timing queues can not be infinite. The timing queue sequence numbers should be reused to represent the time stamps. If the maximum sequence number is M-1, there can only be M timing queues. In order to avoid confusion between RT and RT+M, it should guarantee that the farthest DT of a cell being appended into the timing queues is not larger than RT+(M-1). If the rate of a connection is denoted by ρ, then, the head of line (HOL) cell of the slowest connection can be assigned a DT at most $\frac{1}{\rho_{min}}$ into the future. If only one cell from each connection is kept to stay in the sorting unit, only cells which are at most $\frac{1}{\rho_{min}}$ ahead of the RT should be accommodated in the sorting unit. In order to satisfy the high speed connection performance accuracy, $1/\rho_{max}$ should be chosen as the timing queue granularity. Therefore, in the design of this system, the number of timing queues M should satisfy the inequality below,

$$M \geq \rho_{max}/\rho_{min}. \tag{3.1}$$

There is a shortcoming in the scalability of this architecture. When a broadband network should accept a great number of connections with very wide range of rates, the value M will be very large. With the range of rates increasing, the value of M will increase linearly, which means that the complexity of the implementation of this architecture is

$$O(\rho_{max}/\rho_{min}). \tag{3.2}$$

So this architecture is not suitable for accommodating connections with a very wide range of rates.

3.1.1.2 Rate-Based Grouping Architecture In order to improve the scalability and reduce the complexity of implementation of the Direct Exact Sorting, Rate-Based Grouping Architecture was proposed in [73], as shown in Fig.3.2.

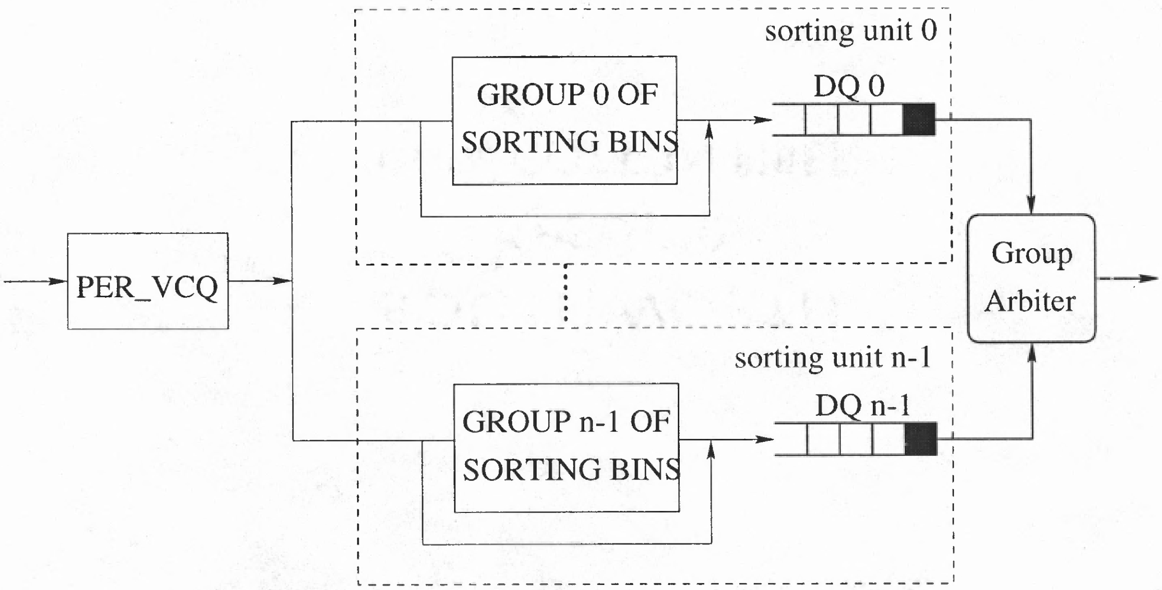


Figure 3.2 Rate-Based Grouping Traffic Shaper

In this scheme, all connections are grouped based on their rates so that each group only needs to shape its connections with a small range of rates. Each group uses a separate shaper that consists of timing queues and a departure queue as shown in Fig. 3.1.

Let μ_i be the granularity parameter of group i (which is used to control the granularity of its group), $\rho_{i,min}, \rho_{i,max}$ be the minimum and maximum connection rates of group i, and n be the number of groups in the whole system. Then, group i can choose its own granularity $g_i = 1/(\mu_i \rho_{i,max})$, so the number of timing queues of group i will be

$$b = \frac{range\ of\ rates}{grain\ of\ each\ timing\ queue} = (\mu_i \rho_{i,max})/(\rho_{i,min}). \tag{3.3}$$

If we divide the rates uniformly, such that for all i, $\rho_{i,max}/\rho_{i,min} = m$, then the total number of timing queues will be $\sum_{i=1}^{n} m\mu_i$. In contrast to the number of timing queues of the Direct Exact Sorting, which is

$$b' = \rho_{max}/\rho_{min} = \prod_{i=1}^{n}(\rho_{i,max}/\rho_{i,min}) = m^n, \tag{3.4}$$

this scheme really reduce the number of timing queues. Because there are more than one group of connections, and each group has its own DQ, there should be a group arbitration logic to arbitrate which group can transmit its HoL conforming cell. Additionally, in order to keep the number of timing queues small enough for low rate connections, large timing queue granularity should be chosen. Consequently, for the low rate connection, the average cell delay increases. The larger the granularity, the worse the performance for connections. The details of the timing queue granularity problem will be explained in the next subsection.

3.1.2 Problems in the Existing Traffic Shaper Architecture

In order to understand the problems in the existing traffic shaper architecture, timing queue granularity should be explained in details. The timing queue granularity means the range of time slots that each timing queue represents. Examples are shown in Fig. 3.3. If the timing queue granularity is chosen as one slot, then kth timing queue represents slot k, and all cells in this timing queue have same DT which is equal to k, and all cells in this timing queue will be appended into the DQ at slot k which is their exact assigned DT. Therefore, it is said that the exact sorting is implemented. If the timing queue granularity is chosen as 100 slots, then *k*th timing queue represents slot k to slot k+99. All cells with DT equal to value between k and k+99 will be appended into this timing queue, and they will be appended into DQ at slot k+99, even one cell with DT equal to k. Consequently, the larger the timing queue granularity, the larger the system delay for the incoming cell.

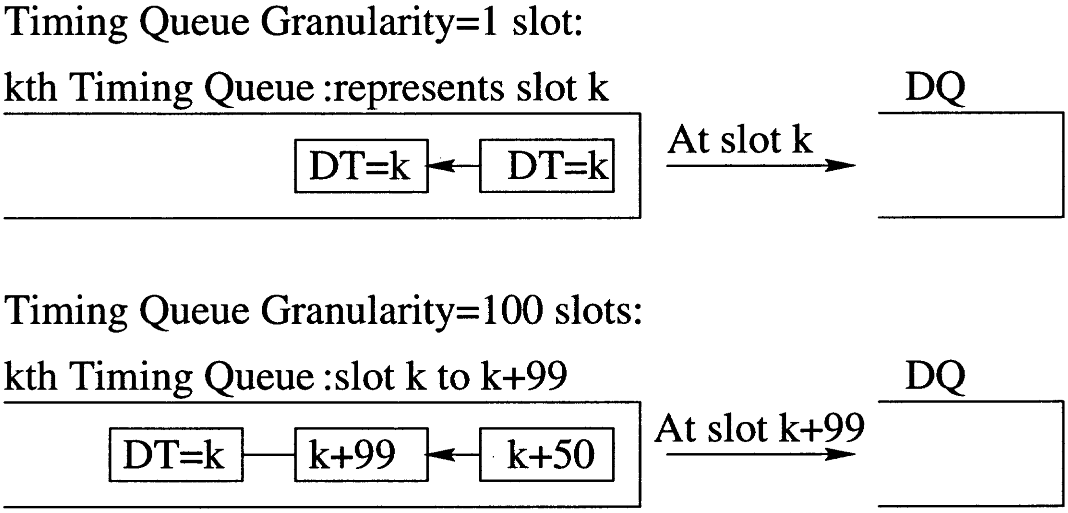

Timing Queue Granularity=1 slot:
kth Timing Queue :represents slot k
DT=k
DT=k
At slot k
DQ
Timing Queue Granularity=100 slots:
kth Timing Queue :slot k to k+99
DT=k
k+99
k+50
At slot k+99
DQ

Figure 3.3 Timing Queue Granularity Depiction

The Direct Exact Sorting architecture implements the exact sorting, and guarantees low system delay for all connections without extra delay due to the coarse sorting. However, it is not scalable to the network with a wide range of the rates distribution because its implementation complexity is

$$O(\rho_{max}/\rho_{min}). \tag{3.5}$$

The Rate-Based Grouping traffic shaper reduces number of timing queues, but the timing queue granularities for the low rate groups are larger than that for the high rate groups. It leads to the performance degradation for connections belonging to the low rate groups. The Rate-Based Grouping architecture reduces the implementation complexity at the price of the performance degradation of the low rate connections. This architecture also requires the implementation of a group arbitrator which increases the implementation complexity. More importantly, this scheme requires a reconfiguration of timing queues and rate groups whenever there is a significant change in overall connection mixture. Otherwise, the performance is degraded substantially.

In order to resolve these problems in the existing traffic shaper architectures, a new traffic shaper architecture with hierarchical timing queues is proposed as shown in Fig. 3.4. It can support individual connection rates from a few kbits/s to several gigabits/s, a 10^6 ratio without the loss of granularity of the departure time (DT). The proposed architecture reduces the number of timing queues greatly without any reduction of performance for all connections. In the proposed architecture, DEDTS algorithm is used to assign DT, which guarantees the multiplexed output stream still satisfying with QoS requirements [95, 94].

3.2 The Architecture Description for the Hierarchical Traffic Shaper

In the proposed architecture, the Per-VCQ structure is still employed at the front of the shaper because Per-VCQ can easily provide flow isolation as well as fair sharing of the link bandwidth among contending connections [23, 10]. Following the Per-VCQ, there is a sorting unit. In this sorting unit, there are two stages of Timing Queues. As discussed in the second section, if only the HoL cell from a VCQ is assigned a DT according to the traffic parameters and the old DT of the most recently departed cell of the same connection, then the possible largest DT is $RT + \frac{1}{\rho_{min}}$, so finite number of timing queues can be used to sort all cells based on their DT. In this scheme, the values of Departure Time (DT) and Real Time (RT) are divided into two parts, as shown in Fig. 3.5.

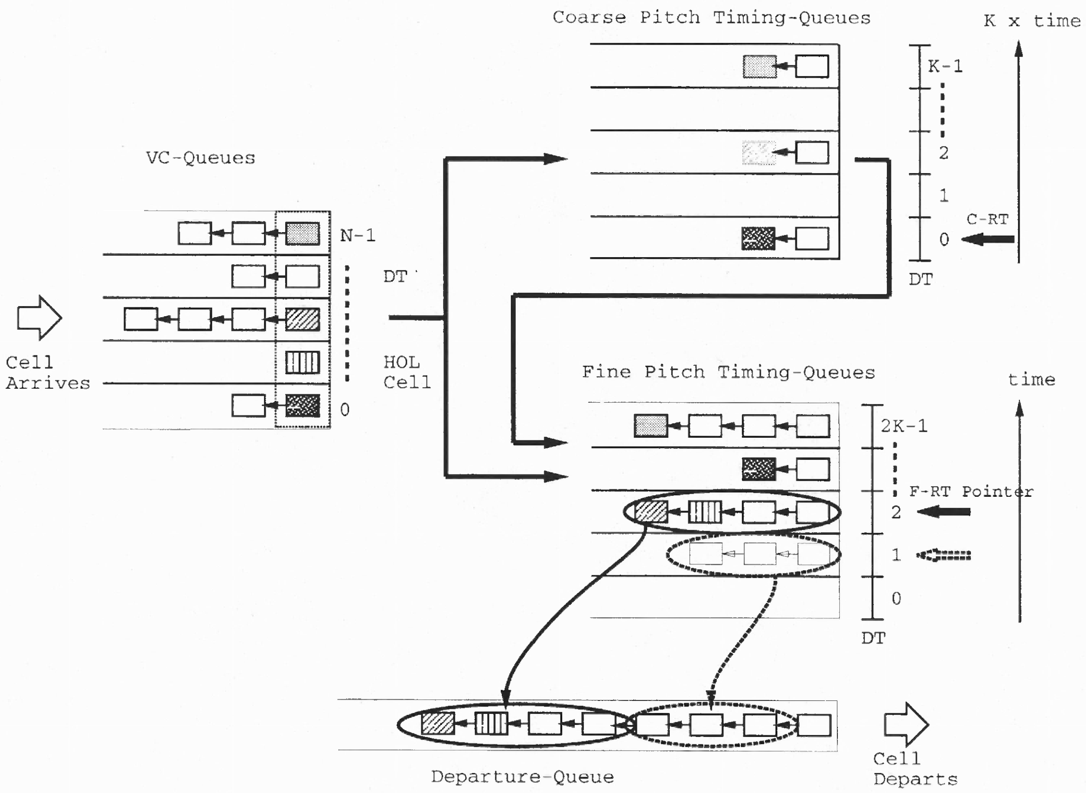


Figure 3.4 Traffic Shaper with Hierarchical Timing Queues

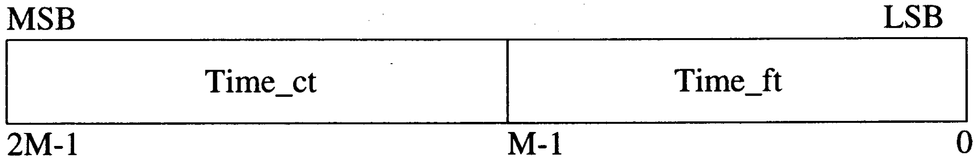
The high part from bit M to bit 2M-1 is called Coarse Pitch Time part, and the low part from bit 0 to bit M-1 is called Fine Pitch Time part, in which, $M = \log_2 K$, and K is the number of timing queues in one set of FPTQ or CPTQ. The DT and can be denoted by (DT_ct, DT_ft), and

$$DT = DT\_ct * 2^K + DT\_ft \tag{3.6}$$

, and the RT can be denoted by (RT_ct, RT_ft), and

$$RT = RT\_ct * 2^K + DT\_ft \tag{3.7}$$

. Similarly, we divide timing queues into two stages. One stage is sequenced by using coarse pitch time value, called Coarse Pitch Timing Queues(CPTQs), which accommodate cells with DT satisfying the inequality: $DT \geq RT + K$. The



Time_ct: Coarse Pitch Timing Part; Time_ft: Fine Pitch Timing Part

Figure 3.5 Data Structure of Time Register

other stage is sequenced by using fine pitch time value, called Fine Pitch Timing Queues(FPTQs), which accommodate cells with DT satisfying the inequality: $DT < RT + K$. Assuming the maximum value of DT_ct=DT_ft=RT_ct=RT_ft=K, then the number of each set of timing queues is K. FPTQs accommodate those eligible HoL cells of VCQs which need to be served in the next K time slots. Each queue of FPTQs is associated with one cell slot. CPTQs store other HoL cells whose DTs are far away from current RT more than K cell slots. The largest sorting range of the two stages of timing queues is

$$\rho_{max}/\rho_{min} \leq K^2 \qquad (3.8)$$

, so the total number of the timing queues will be

$$b \geq 2\sqrt{\rho_{max}/\rho_{min}} \qquad (3.9)$$

when ρ_{max}/ρ_{min} is given. However, b has to be at least equal to $3K$ for the correct operation of our sorting unit. We will explain it later.

Although in each CPTQ, all cells have same DT_ct, they are out of order due to their different DT_ft. In order to sort the cells in the CPTQs, all cells will be appended to FPTQs based on their DT_ft. In every cell slot, the HoL cell of the CPTQ pointed by the coarse pitch real time (RT_ct) is appended into the corresponding FPTQ based on the cell's DT_ft. As the real time advances, linked list of cells in the FPTQ pointed by fine pitch real time RT_ft pointer are appended into a FIFO Departure Queue(DQ) where they are served at the line rate. In order to sort cells correctly while they are being appended from the CPTQs to the FPTQs, a prefetching scheme is employed. Correct sorting of these appended cells requires to employ $2K$ FPTQs when the maximum value of DT_ft is K. Hence, the total number of timing queues that guarantees the correct sorting of cells based on their DTs is

$$b \geq 3\sqrt{\rho_{max}/\rho_{min}} \tag{3.10}$$

. This ensures that while the one half of FPTQs are being served, only the other half are appended with cells from CPTQs. These newly appended cells do *not fall behind the fine pitch real time pointer.* The process is shown in Fig. 3.6. One set of FPTQs accommodate cells with $DT\_ct = RT\_ct$, and another one set accommodate cells with $DT\_ct = RT\_ct + 1$.

The real implementation of the prefetching scheme is shown in Fig. 3.7. One set of FPTQs are used to accommodate cells with even DT_ct (called even FPTQs' unit), and another set of FPTQs are used to accommodate cells with odd DT_ct (called odd FPTQs' unit). The correctness will be shown in an example later.

3.3 The Algorithm Description for the Hierarchical Traffic Shaper

In our scheme, the Departure Event Driven Traffic Shaping Algorithm (DEDTS) is employed. In the whole shaping process, two events trigger a series of operation for cells' moving from VCQ to CPTQ or FPTQ. They are one cell's arrival and one cell's departure. At least one cell is moved from CPTQ to FPTQ in every cell slot if the CPTQ pointed by RT_ct+1 is not empty.

The number of queues in FPTQs and in CPTQs are 2K and K, respectively. Cells in CPTQs with even DT_ct will be appended to FPTQ[DT_ft], and others will be appended to FPTQ[$DT\_ft + K$] once their $DT\_ct = RT\_ct + 1$.

1. Fig. 3.8 describes the operations performed when a cell arrives.

When a cell with Connection ID (CID) i arrived, if there is a cell with the same CID in the sorting unit, the new incoming cell is appended to its VCQ[i]. Otherwise, it is assigned a DT. If the $DT\_ct \neq RT\_ct$ or $RT\_ct + 1$, it is appended to CPTQ[DT_ct]. Otherwise it is appended to FPTQ[DT_ft] if DT_ct is even or appended to FPTQ[DT_ft+K] if DT_ct is odd.

2. When a cell leaves, the process described in Fig. 3.9 is followed.

   When a cell with CID i departs from DQ, a new DT for the next cell with the same CID is calculated using DEDTS. If the VCQ[i] is empty, the new DT is stored for the future incoming cell. Otherwise, the HoL cell of VCQ[i] is assigned the new DT. Based on the value of DT, the cell is appended to CPTQ or FPTQ same as the process triggered by the arrival event.

3. In each cell slot, the state of FPTQ[RT_ft] is checked. If it is not empty, all cells in this queue are appended to DQ, and the HoL cell of DQ is served. Meanwhile, the state of CPTQ[$RT\_ct + 1$] needs to be checked. If it is not empty, its HoL cell should be appended into FPTQ[DT_ft] or FPTQ[DT_ft+K] depending on the $DT\_ct + 1$ even or odd, where the DT_ft is the fine departure time of the cell being moved.

   When RT_ct is about to be incremented to $RT\_ct + 1$, if there are unserved cells in CPTQ[RT_ct+1], they will be all appended to either FPTQ[K-1] or FPTQ[2K-1] depending on RT_ct+1 being even or odd.

An example of this process is shown in Fig. 3.10, Fig. 3.11, Fig. 3.12, Fig. 3.13, and Fig. 3.14. In this example, there are four connections, and $\rho_{max}/\rho_{min} = 16$. Based on the algorithm described before, K is equal to 4, and in FPTQs' unit, there are 8 timing queues, and in CPTQs' unit, there are 4 timing queues. FPTQ[0] to FPTQ[3] are the even FPTQs', and FPTQ[4] to FPTQ[7] are odd FPTQs' unit.

In Fig. 3.10 (cell slot 0), cells in FPTQ[0] pointed by RT_ft=0 are appended to DQ, and the HoL cell of DQ that belongs to connection 0 is served. Then, the HoL cell of connection 0 is assigned a new Departure Time (DT), which is a binary value 0110. Because its DT_ct=01 which is equal to RT_ct+1=00, and its DT_ft=10(Binary), it is appended to FPTQ[2+K]=FPTQ[2+4]=FPTQ[6]. In the same slot, CPTQ[RT_ct+1]=CPTQ[1] is checked. It is not empty, so its HoL cell is inserted to odd FPTQs' unit. Since its DT_ft=11(Binary), it is appended to FPTQ[3+K]=FPTQ[3+4]=FPTQ[7].

In Fig. 3.11 (cell slot 1), cells in the FPTQ[1] pointed by RT_ft=1 are appended to DQ, and the HoL cell of DQ is served. This cell's Connection ID (CID) is 3, so the HoL cell of connection 3 is assigned a new DT, say 1000. Its DT_ct =10(Binary), is unequal to RT_ct or RT_ct+1, so it is appended to CPTQ[DT_ct]=CPTQ[2]. Meanwhile, the HoL cell of CPTQ[1] is appended to odd FPTQs' unit FPTQ[0+K]=FPTQ[4] since its DT_ft=0 and DT_ct=1.

In Fig. 3.12 (cell slot 2), RT_ft=2 is pointing to an empty FPTQ[2], and DQ is also empty, so there is no cell which needs to be served. But there is a new cell with CID=0 arriving. Because there is another cell with same CID in the sorting unit, the new cell can not be assigned a DT, so it is appended to VCQ[0].

Similarly, in Fig. 3.13 (cell slot 3), a new cell with CID 1 is received, and append it to VCQ[1]. In Fig. 3.14 (cell slot 4), as RT_ct increases from 0 to 1, we begin to serve odd FPTQ unit, and append all cells in FPTQ[4] to DQ, and serve the HoL cell of DQ whose CID is 2. Then, a new DT is assigned to HoL of connection 2. Because the new DT=0101 (binary) with a *DT_ct* = 01 = *RT_ct*, it should be appended to current served FPTQs' unit FPTQ[1+K]=FPTQ[5]. In the same cell slot, it finds that CPTQ[RT_ct+1]=CPTQ[2] is not empty, so its HoL cell is appended to FPTQ[0] since the cell's DT_ft=0. These five slot operations show that all cells' time stamps in FPTQs' unit are sorted successfully although they have a wrong order in CPTQs' unit originally. This example also shows that two sets of FPTQs are needed to prefetch and sort time stamps correctly.

This prefetching scheme guarantees that all cells with the same *DT_ct* have been ready in the FPTQs to be served just before *RT_ct* is incremented by one to become this *DT_ct* , and improves the efficiency for this architecture. This architecture only needs $b \geq 3\sqrt{\rho_{max}/\rho_{min}}$ timing queues to satisfy the exact sorting, which is much less than the number of timing queues in the Direct Exact Sorting Traffic Shaping architecture. It doesn't need complex logic arbitration as in the case of grouping architecture, but still implements the exact sorting of all time stamps, and keeps the performance for each connection as good as in the Direct Exact Sorting architecture.

3.4 Performance Evaluation for the Hierarchical Traffic Shaper

In this section, the performances of the traffic shaper with Hierarchical Timing Queues, the traffic shaper with the Direct Exact Sorting algorithm, and the traffic shaper with Rate-Based Grouping algorithm are compared. The objective of this study is to demonstrate that the proposed traffic shaper can reduce the implementation complexity while keeping good performance for all connections. The performance metrics used in this evaluation is the average cell delay for all three cases.

In the simulation experiments, Line Bit Rate LBR is chosen as 155.52Mbps, and $\rho_{max}/\rho_{min} = 10^6$, so the Direct Exact Sorting Traffic Shaper needs $\rho_{max}/\rho_{min} = 10^6$ timing queues to work properly, while our traffic shaper with Hierarchical Timing Queues only need $3\sqrt{\rho_{max}/\rho_{min}} = 3000$ timing queues to get the same performance as the Direct Exact Sorting Traffic Shaper. For the Rate-Based Grouping Traffic Shaper, in the experiment, all connections are divided into two groups. First group contains high speed connections with rates from $10^3\rho_{min}$ to ρ_{max}. Second group contains low speed connections, rates are from ρ_{min} to $10^3\rho_{min}$. As discussed in the section 2, the total number of sorting bins is $\sum_{i=1}^{n} m\mu_i$, and in this experiment, $m = 10^3$, $\mu_0 = 1$, and $\mu_1 = 2$. Therefore, for the group of high rate connections, the granularity is 1, which is as same as the exact sorting scheme. For the group of low rate connections, the granularity is $m/\mu_1 = 500$, which is 500 times worse than that of the exact sorting scheme, and introduces large inaccuracy for low rate connections as shown in the simulation result of Fig. 3.15. Consequently, the total number of sorting queues in the Rate-Based Grouping Traffic Shaper is also 3000, same as the Hierarchical Timing Queues Traffic Shaper.

In the experiment, 8 VBR sources with diverse rate distribution are used, and the grouping parameters described above are chosen as:

$VBR0 : PCR = 0.68LBR = 250kcells/sec, MBS = 500;$

$VBR1 : PCR = 0.27LBR = 100kcells/sec, MBS = 200;$

$VBR2 : PCR = 0.14LBR = 50kcells/sec, MBS = 100;$

$VBR3 : PCR = 0.03LBR = 10kcells/sec, MBS = 50;$

$VBR4 : PCR = 0.003LBR = 1kcells/sec, MBS = 40;$

$VBR5 : PCR = 0.0003LBR = 100cells/sec, MBS = 20;$

$VBR6 : PCR = 0.00003LBR = 10cells/sec, MBS = 10;$

$VBR7 : PCR = 0.000003LBR = 1cells/sec, MBS = 5.$

Fig. 3.15 (a) shows the average shaping delay of high rate connection with PCR=50k cells/sec, and Fig. 3.15 (b) shows the average shaping delay of low rate connection with PCR=100 cells/sec.

Fig. 3.15(a) shows that all the three models have the almost identical performance for the high rate connections because all of them choose the same timing queue granularity. While, for the low rate connections, Fig. 3.15(b) shows that the traffic shaper with Hierarchical Timing Queues and the traffic shaper with Direct Exact Sorting algorithm have almost same performance, but the Rate-Based

Grouping traffic shaper has much worse performance whose average delay for the low rate groups is more than twice larger than that in Direct Exact Sorting and Hierarchical Timing Queues traffic shaper. The reason is that the Rate-Based Grouping traffic shaper chooses the large granularity for the low rate connections. The larger the granularity, the worse the performance. In experiments, it finds that the performance of connections in the Rate-Based Grouping traffic shaper depends on the traffic characteristic. When the traffic falls in groups with large granularity, its performance will degrade.

The simulation experiments show that the traffic shaper with hierarchical timing queues architecture keeps the performance as good as the traffic shaper with direct exact sorting algorithm while it reduces the implementation complexity greatly. Compared with the traffic shaper with direct exact sorting architecture, the hierarchical timing queues architecture reduces the number of timing queues substantially. Compared with the rate-based grouping traffic shaper, it avoids the implementation of the complex scheduler, and keeps the comparable number of timing queues. For the rate-based grouping traffic shaper, although it also reduces the number of sorting queues, its performance for low rate connection degrades greatly. It requires hardware reconfigurations since the number of groups depends on the continuously changing traffic distribution. But, for the traffic shaper with hierarchical timing queues, although it reduces the implementation complexity greatly, it still maintains the performance for all connections as good as the traffic shaper with direct exact sorting architecture. Its simplicity of implementation, good performance for very diverse traffic parameters, and implementation structure independent on the traffic distribution make it more practical and attractive for ATM switch designers.

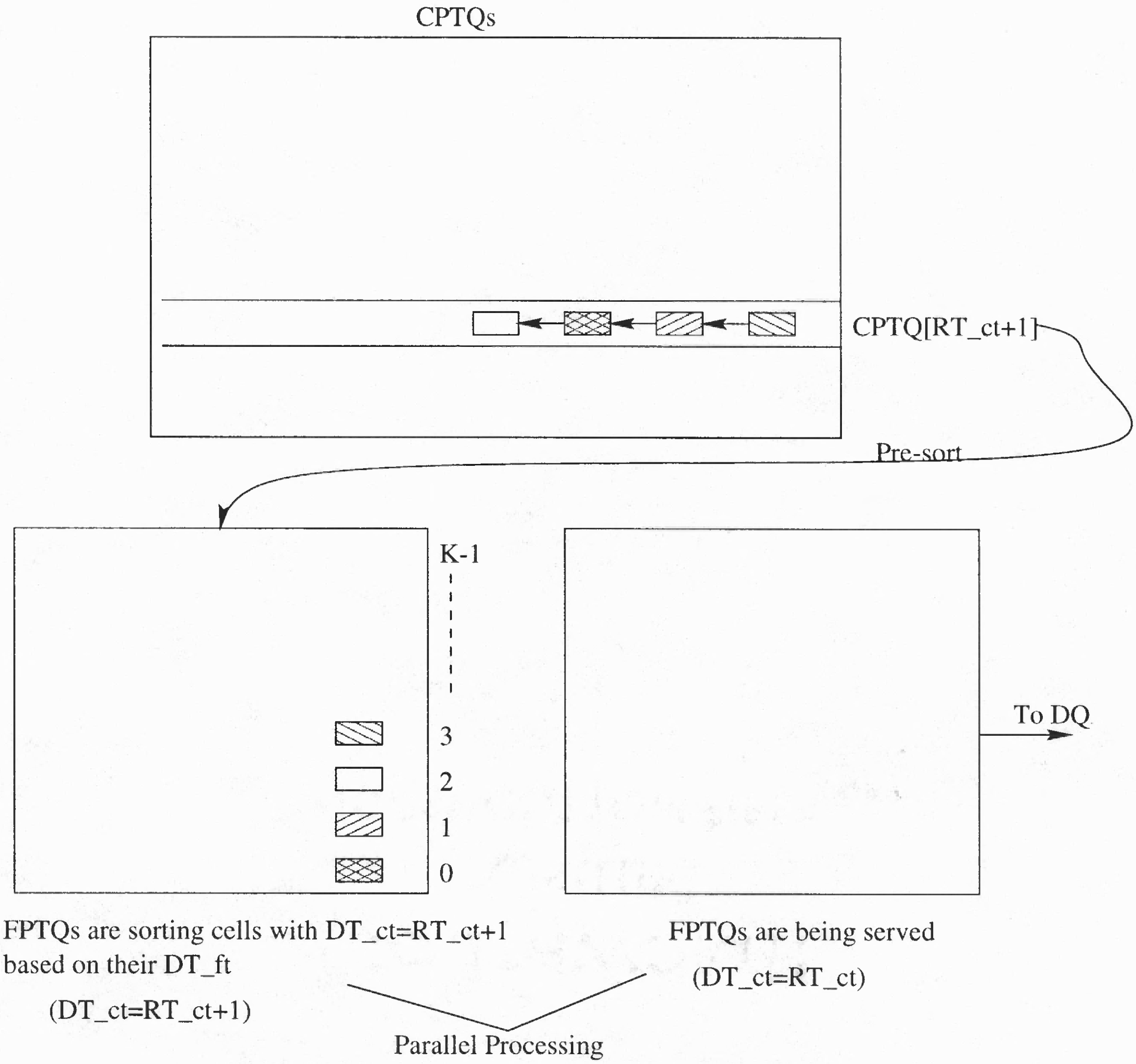


Figure 3.6 The Process of the Prefetching Scheme

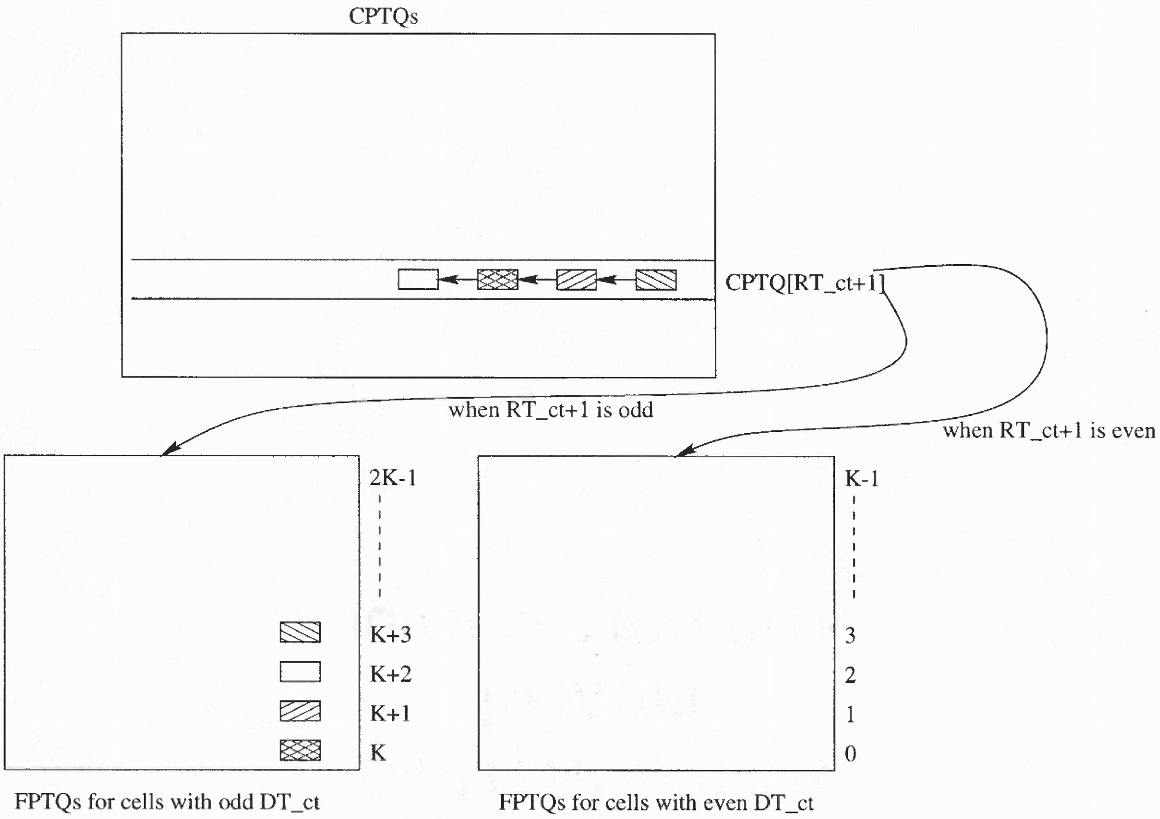


Figure 3.7 The Real Implementation of the Prefetching Scheme

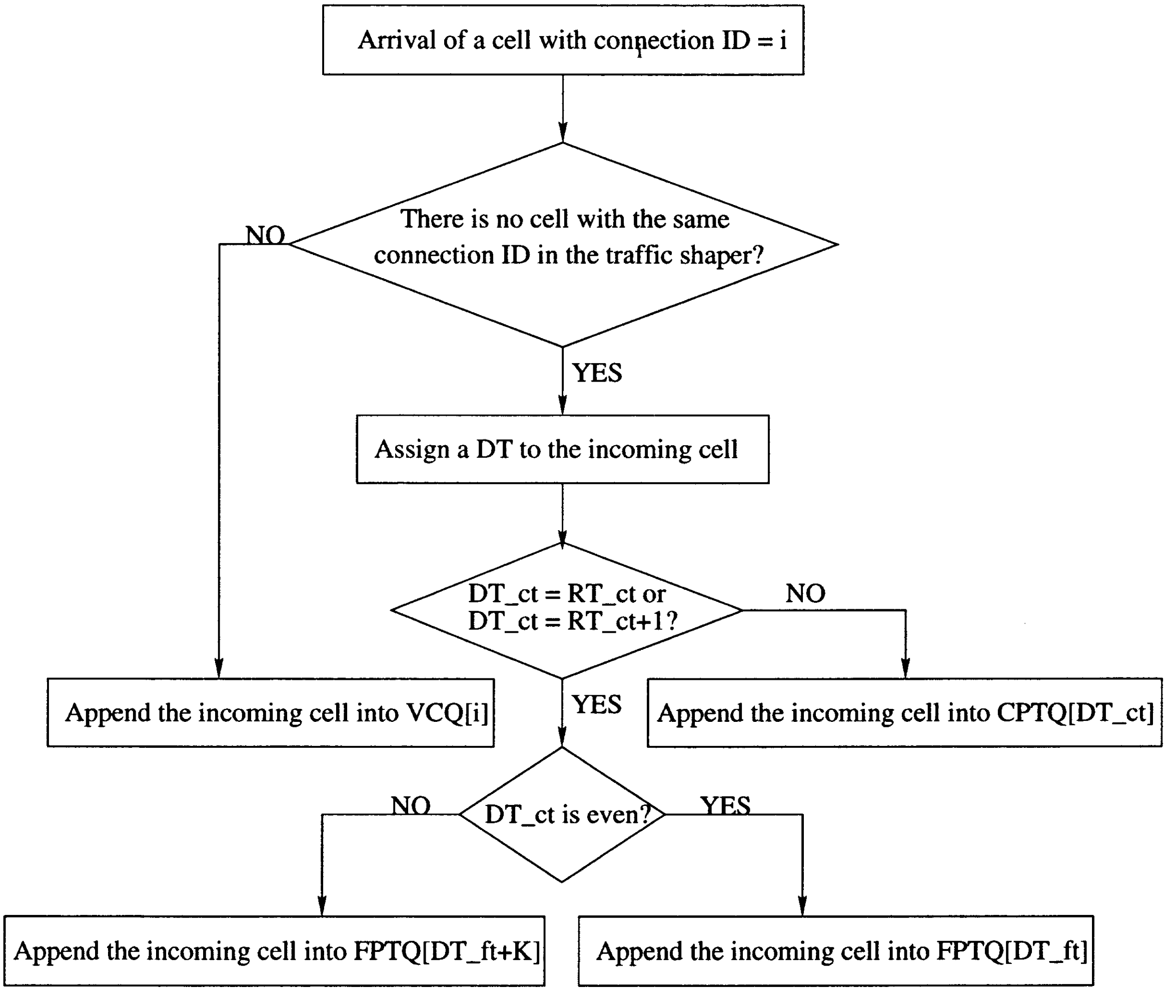

Arrival of a cell with connection ID = i
There is no cell with the same connection ID in the traffic shaper?
NO
YES
Assign a DT to the incoming cell
DT_ct = RT_ct or DT_ct = RT_ct+1?
NO
YES
Append the incoming cell into VCQ[i]
Append the incoming cell into CPTQ[DT_ct]
DT_ct is even?
NO
YES
Append the incoming cell into FPTQ[DT_ft+K]
Append the incoming cell into FPTQ[DT_ft]

Figure 3.8 The Process Triggered by the Arrival of A Cell with the Connection ID i

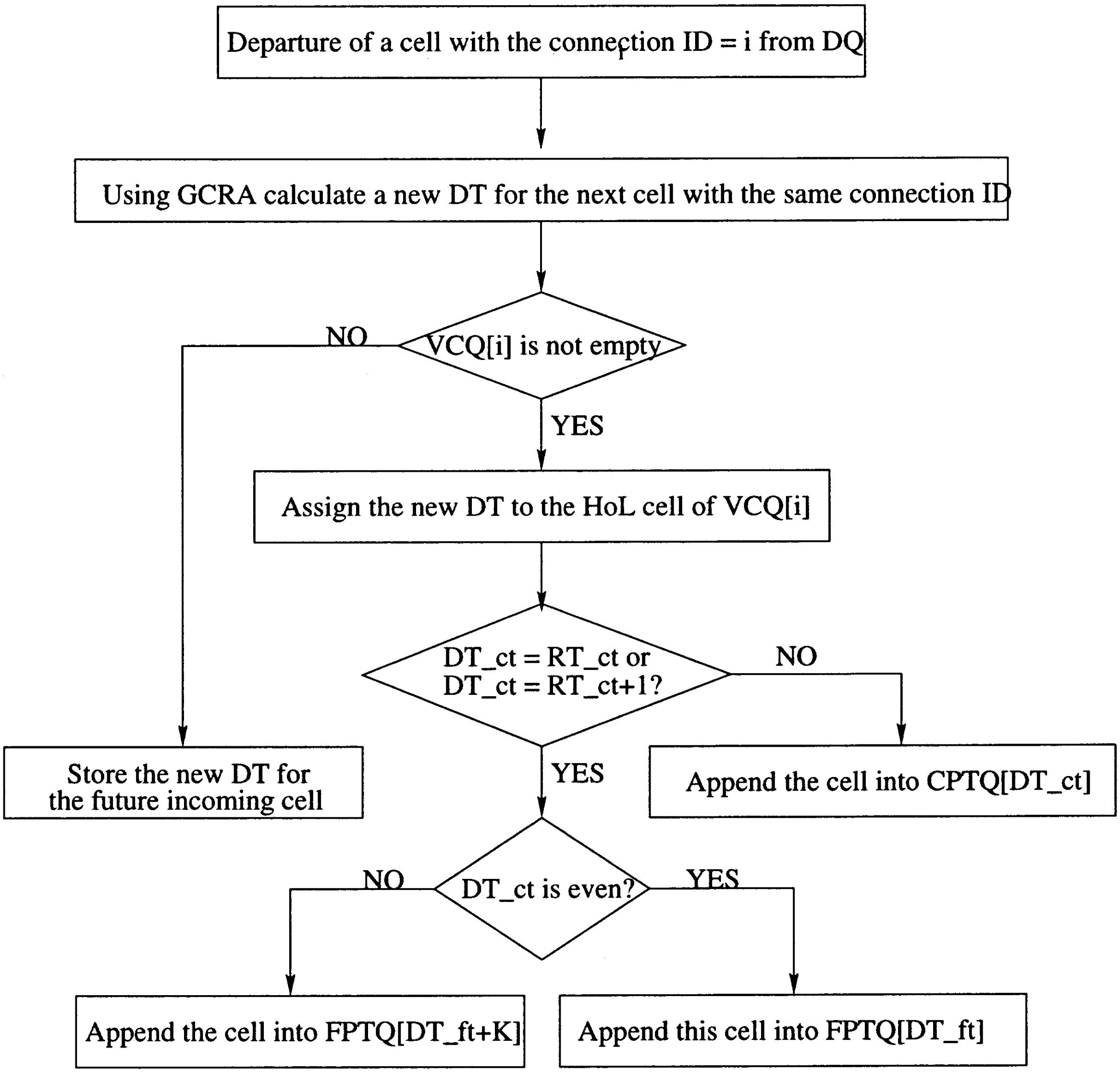
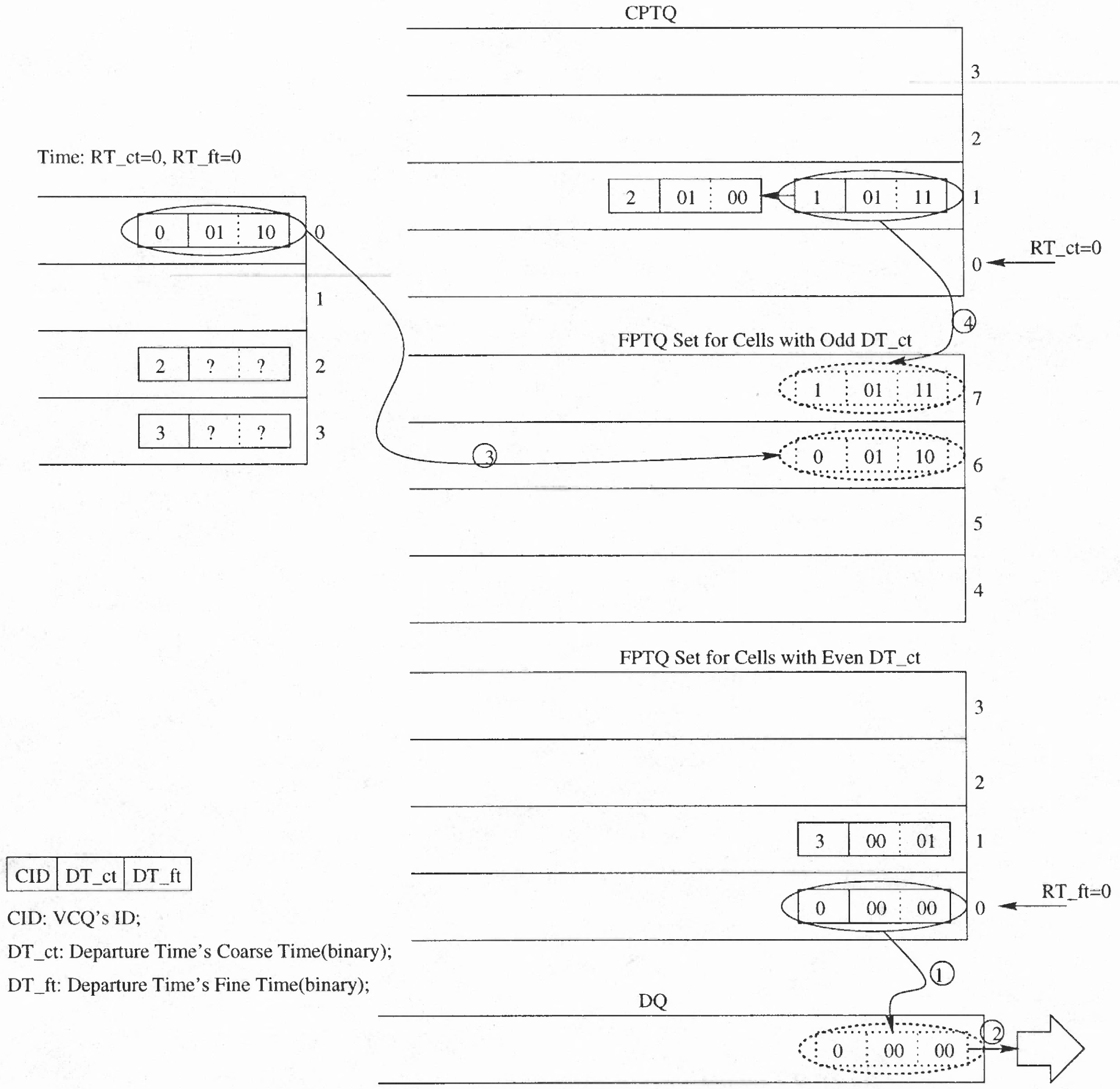

Departure of a cell with the connection ID = i from DQ
Using GCRA calculate a new DT for the next cell with the same connection ID
VCQ[i] is not empty
NO
YES
Assign the new DT to the HoL cell of VCQ[i]
DT_ct = RT_ct or DT_ct = RT_ct+1?
NO
YES
Store the new DT for the future incoming cell
Append the cell into CPTQ[DT_ct]
DT_ct is even?
NO
YES
Append the cell into FPTQ[DT_ft+K]
Append this cell into FPTQ[DT_ft]

Figure 3.9 The Process Triggered by the Departure of A Cell with the Connection ID i from DQ



CID: VCQ's ID;

DT_ct: Departure Time's Coarse Time(binary);

DT_ft: Departure Time's Fine Time(binary);

Figure 3.10 The Process of Moving Cells between VCQ, CPTQ, FPTQ (Slot 0)

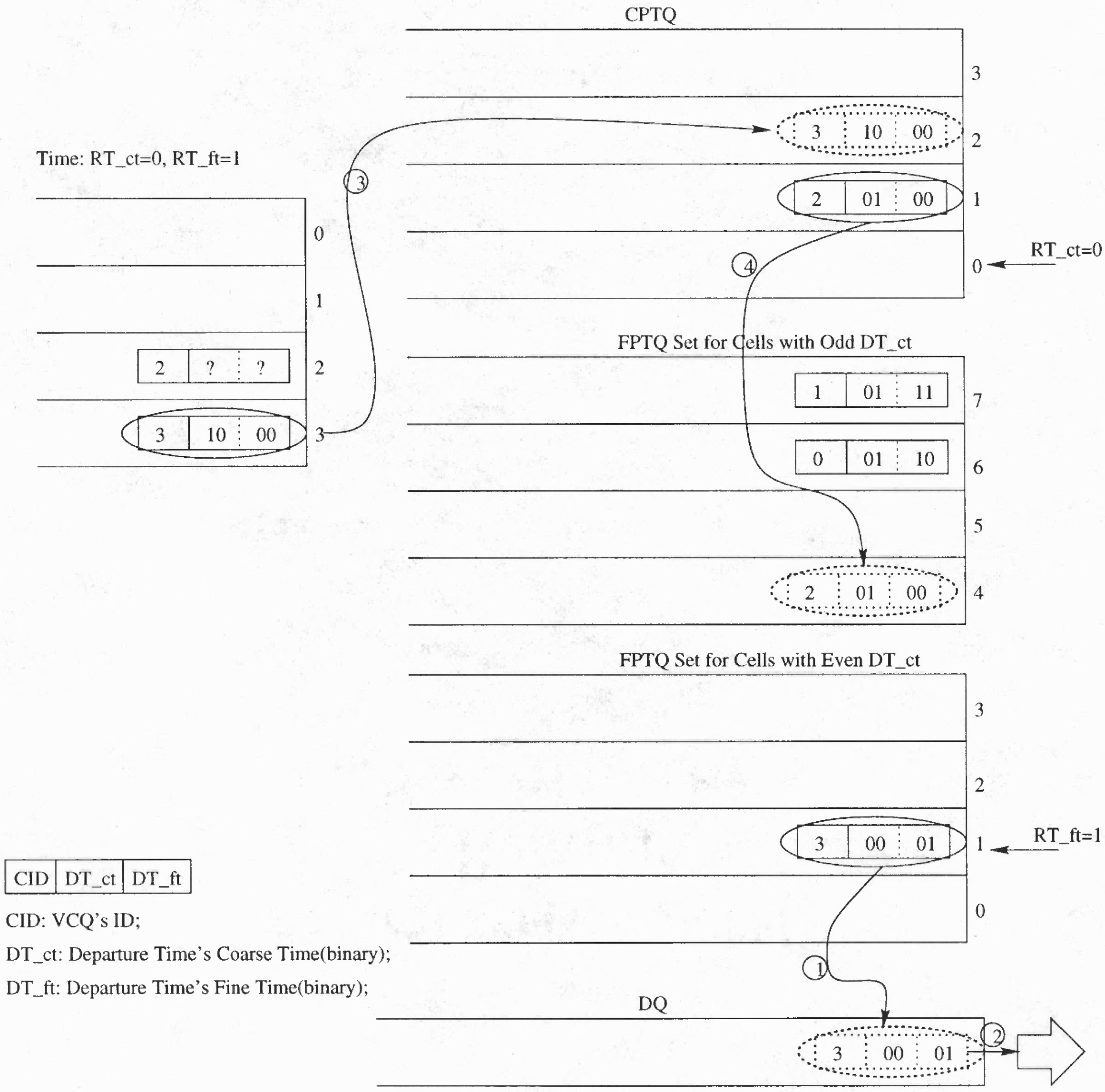

CPTQ
Time: RT_ct=0, RT_ft=1
3
3 10 00
2
2 01 00
1
RT_ct=0
0
4
0
1
2 ? ?
2
3 10 00
3
FPTQ Set for Cells with Odd DT_ct
1 01 11
7
0 01 10
6
5
2 01 00
4
FPTQ Set for Cells with Even DT_ct
3
2
3 00 01
1
RT_ft=1
0
CID DT_ct DT_ft
CID: VCQ's ID;
DT_ct: Departure Time's Coarse Time(binary);
DT_ft: Departure Time's Fine Time(binary);
1
DQ
3 00 01
2

Figure 3.11 The Process of Moving Cells between VCQ, CPTQ, FPTQ (Slot 1)

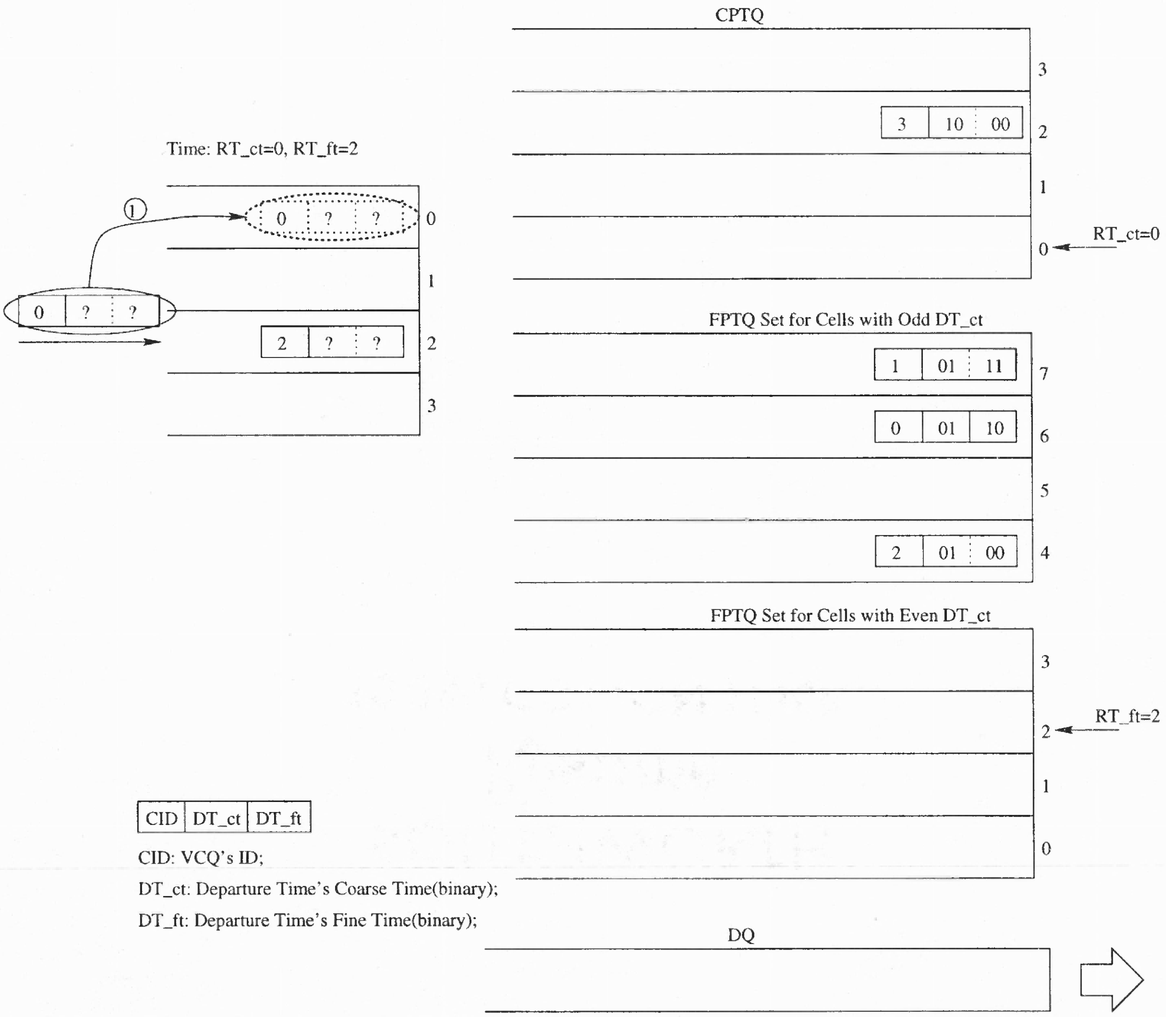
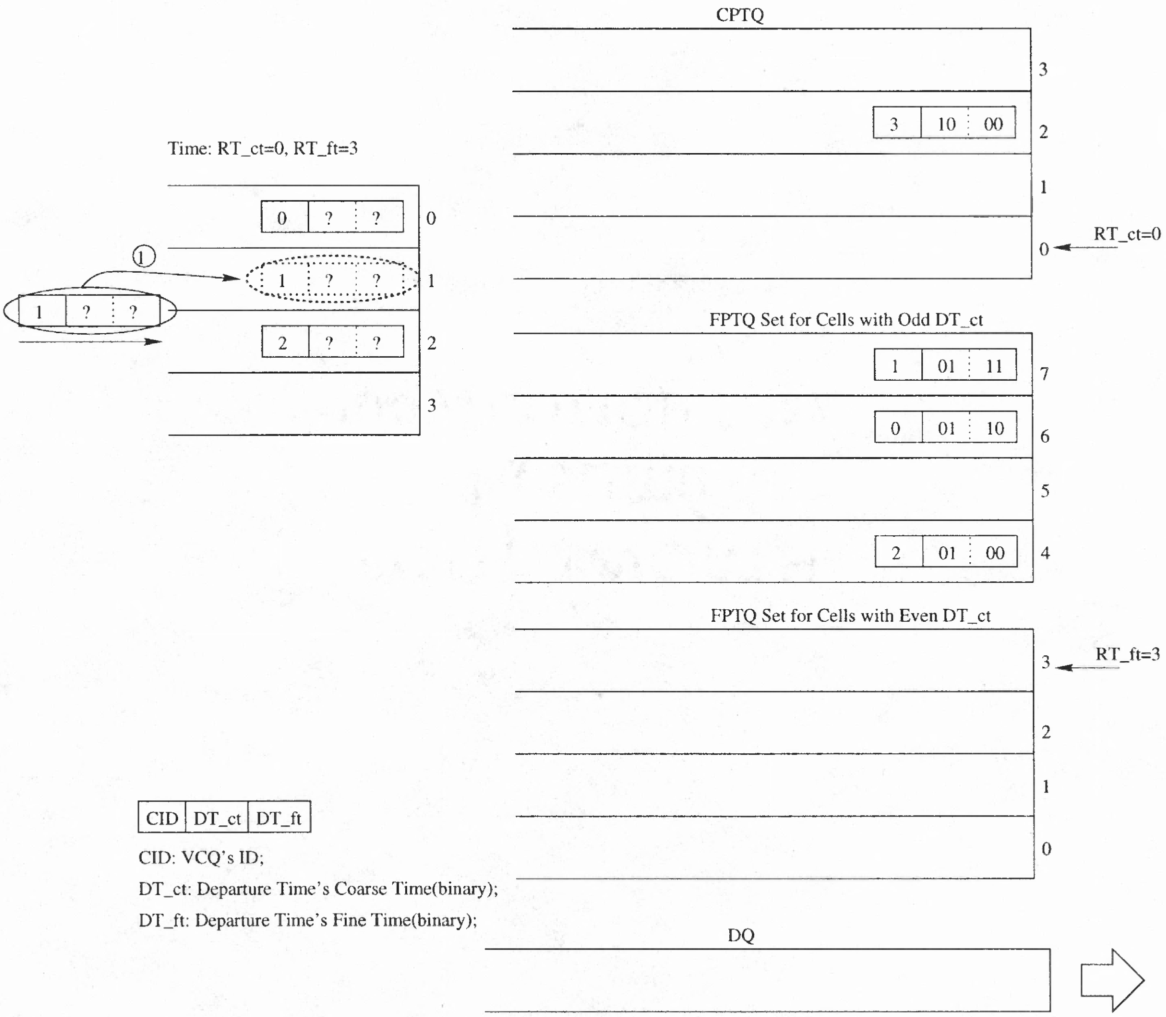


Figure 3.12 The Process of Moving Cells between VCQ, CPTQ, FPTQ (Slot 2)



CID: VCQ's ID;

DT_ct: Departure Time's Coarse Time(binary);

DT_ft: Departure Time's Fine Time(binary);

Figure 3.13 The Process of Moving Cells between VCQ, CPTQ, FPTQ (Slot 3)

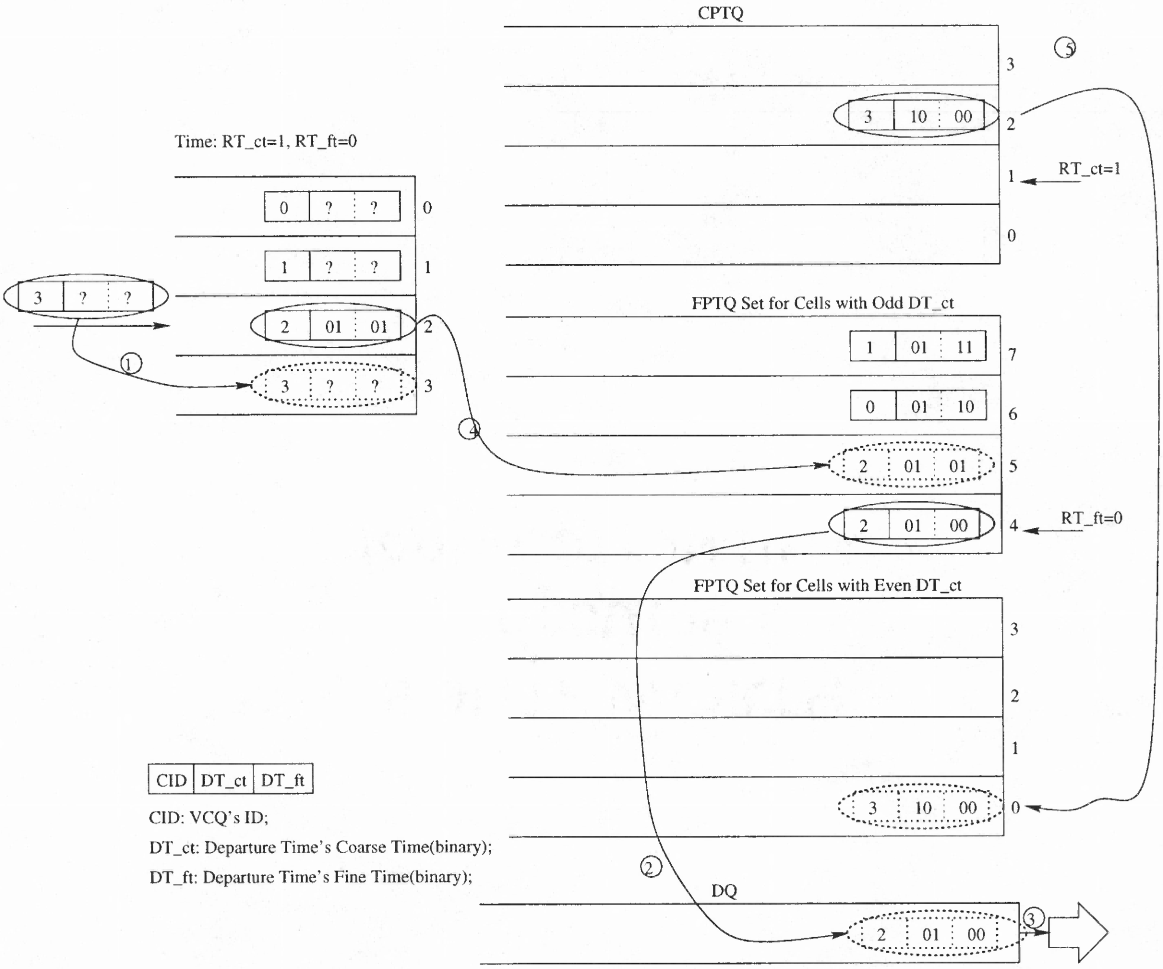


Figure 3.14 The Process of Moving Cells between VCQ, CPTQ, FPTQ (Slot 4)

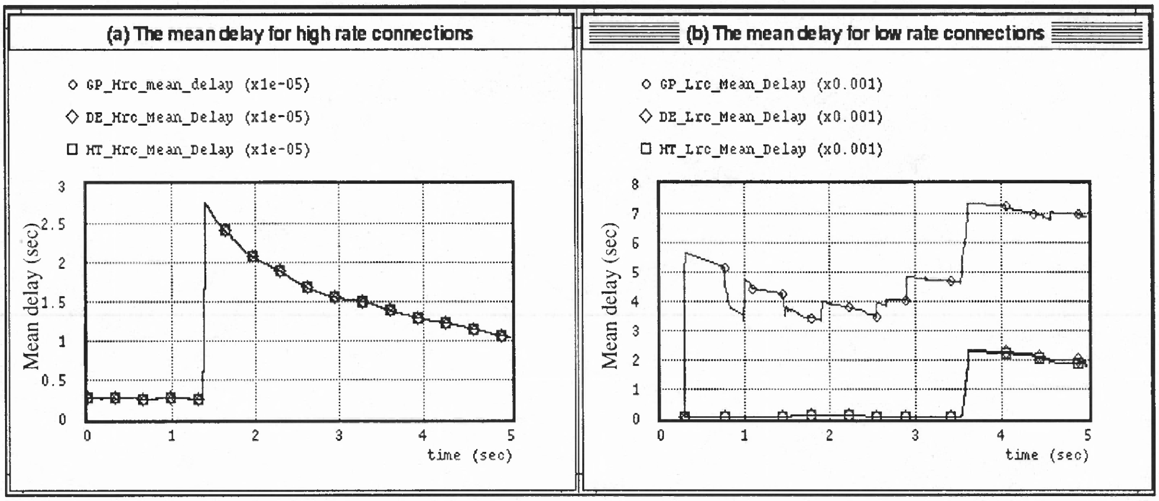


Figure 3.15 Mean Cell Delay for these three schemes

CHAPTER 4

THE RATE-ADAPTIVE SCHEDULER IMPLEMENTED BY HIERARCHICAL SORTING UNITS WITH LOOKAHEAD ALGORITHM

4.1 Introduction

A high speed packet switch should be capable of satisfying QoS requirements of each kind of services while it is being fair in the bandwidth assignment. Cell Transfer Delay (CTD) and Cell Delay Variation (CDV) are two of most important issues related to QoS requirements. In order to reduce CTD and CDV while keeping the fairness in the bandwidth assignment, many scheduling schemes are proposed [2, 14, 35, 69, 70, 74, 78, 83]. FCFS is the simplest scheduler scheme, and is implemented widely. But it doesn't consider CDV and the fairness of the bandwidth assignment [84]. Round Robin(RR) takes the service fairness into consideration, but when all connections do not have the same bandwidth requirements, RR method becomes unfair [20]. Weighted round-robin (WRR) resolves the unfairness in RR [74, 87], but it doesn't consider CDV which can lead to high cell violation at User Parameter Control(UPC). In [20], scheduler schemes FCFS, RR, EDF, WRR and LDBE are compared, and LDBE is proved to be the best algorithm. Its essence is to use GCRA to assign DT to each cell, and schedule cells based on their DT order. In this paper, it is named as GCRA scheduler. Using GCRA, the scheduler can satisfy the QoS requirements for various traffic and support a fair bandwidth assignment [20].

In this case, the scheduler does not care whether a cell is due or overdue, it sorts cells according to their DTs, and sends them one by one starting with the lowest DT. Fast and efficient sorting of these DTs is the bottleneck in this scheduler, especially, when the number of connections is very large, which makes it impractical in the real system implementation. For the proposed hierarchical sorting unit, we can implement the fast sorting efficiently. In this section, the algorithm described in the previous section is redesigned so that the hierarchical sorting unit works like a rate-adaptive GCRA scheduler. When the traffic is light, it can work like a GCRA scheduler to reduce the CTD as much as possible. When the traffic is heavy, it can function as a shaper to shape the input traffic to avoid the congestion on the outlink [94, 99]. It is named as Hierarchical Rate-Adaptive Scheduler (HRAS). From the implementation viewpoint, it is a practical scheme.

4.2 Algorithm Description of the Hierarchical Rate-Adaptive GCRA Scheduler with Lookahead Algorithm

The proposed hierarchical sorting unit can naturally realize the sorting of DTs which is calculated by GCRA. When it acts as a scheduler, it should avoid the starvation of the outlink. Consequently, when the current FPTQ which is pointed by the RT_ft pointer is empty, *Lookahead* algorithm can be used to find the nearest nonempty FPTQ and catch the cell with the smallest DT in FPTQs' unit. By using the lookahead algorithm in the hierarchical sorting units, it can be close to an ideal GCRA scheduler.

For practical hardware implementation, it is impossible to lookahead too deep in one cell slot. The more FPTQs to lookahead, the more the memory access operations in one cell slot, which is bounded by the memory technology used. Assuming n to be the largest number of FPTQs that can be lookaheaded in one cell slot, then in the limited n steps, two ways are used to improve the efficiency of the lookahead algorithm so that its performance can approach to the ideal GCRA scheduler which can essentially be considered as a scheduler with an unlimited lookahead depth in one cell slot.

One way is to avoid pulling the lookahead pointer back. A direct design of the lookahead scheme is to lookahead from the current FPTQ, and continue to check successive n FPTQs in each lookahead action. If one nonempty FPTQ is found, the lookahead action ends. This operation limits the range the lookahead pointer can reach. If the nonempty FPTQ is far away from the one pointed by RT_ft, it can not be reached quickly. In order to "visit" more FPTQs in a short time, this scheme avoids to pull the lookahead pointer back to the current FPTQ pointed by RT_ft. After each lookahead action, it is kept where it is, then, in the next slot, another lookahead action begins from this position. The process is shown in Fig. 4.1.

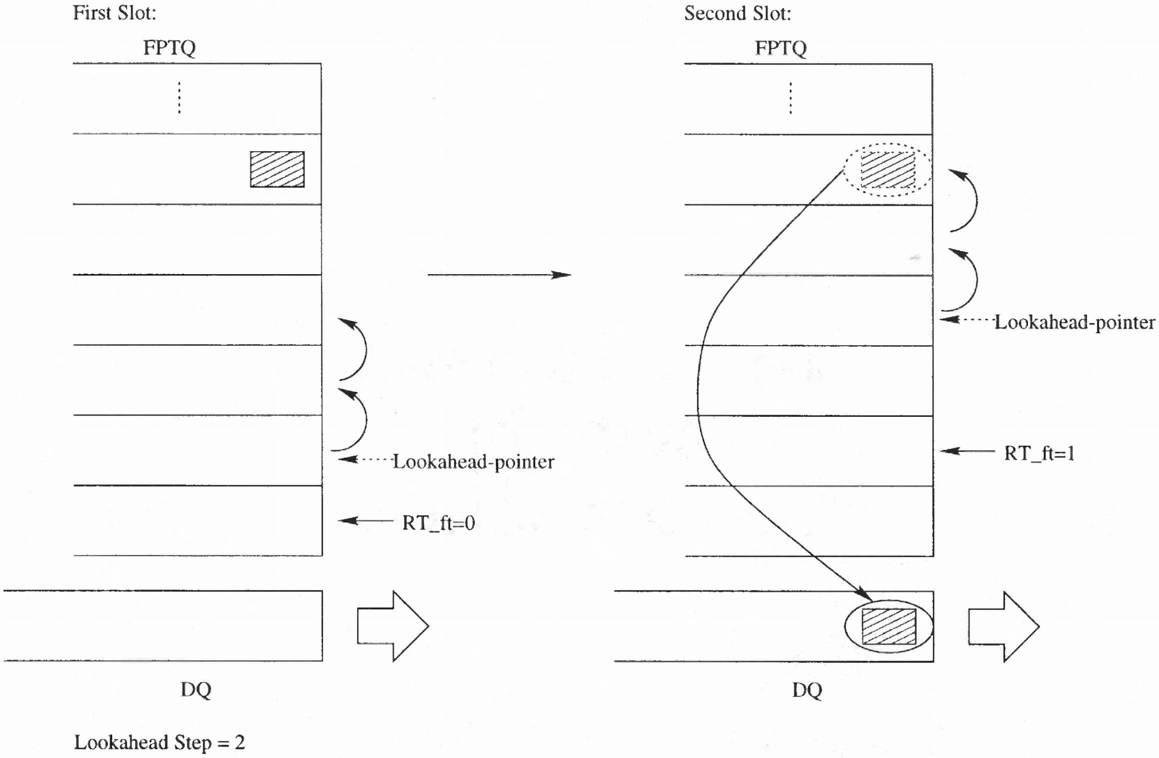


Figure 4.1 An Example of Moving the Lookahead Pointer

In each lookahead action, although the lookahead pointer just moves at most n steps, it finally can lookahead far away from the FPTQ pointed by the current RT_ft.

When a cell is appended into FPTQs from either Per-VCQs or CPTQs, its position is checked. If it is appended to a FPTQ whose position is higher than the real time pointer RT_ft, but lower than the lookahead pointer, which means this is the nearest nonempty FPTQ, so this cell will be appended to DQ directly. By allowing this operation, an inaccuracy of time stamp sorting may be led in DQ. In order to reduce the chance of the inaccuracy while maintaining the outlink utilization, a threshold is set to the length of DQ. Only when the length of DQ is shorter than the threshold, the operation explained above will be performed, otherwise, the lookahead pointer is backuped to the FPTQ where the cell is appended to. An example is shown in Fig. 4.2. In this example, a new incoming cell is appended to the FPTQ which is located between RT_ft and Lookahead-pointer, and the length of DQ reaches the threshold which is equal to 1, the Lookahead-pointer has to be pulled back to the FPTQ where the cell is appended.

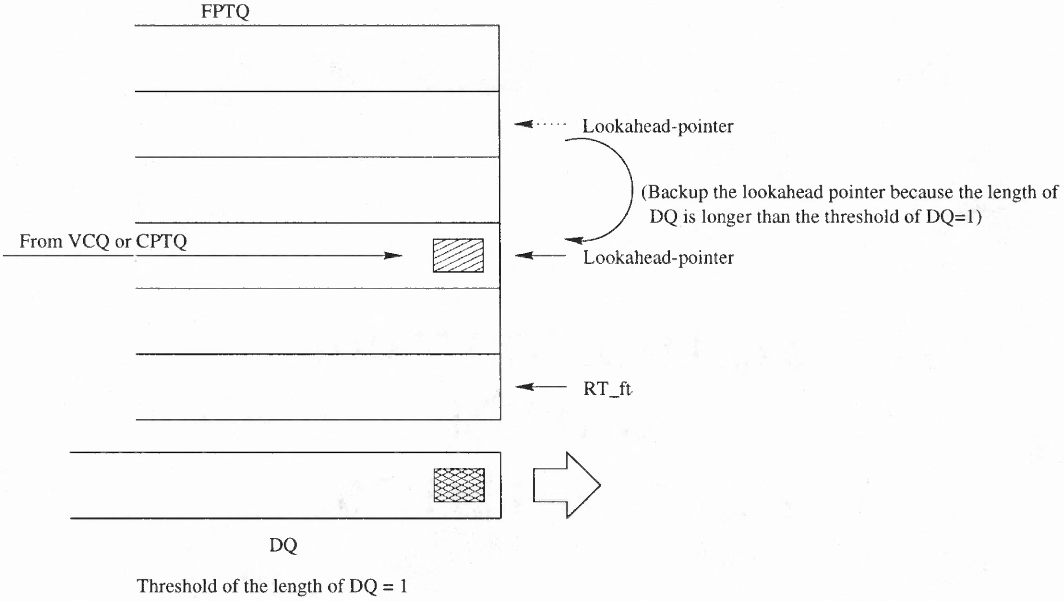

FPTQ
Lookahead-pointer
(Backup the lookahead pointer because the length of DQ is longer than the threshold of DQ=1)
From VCQ or CPTQ
Lookahead-pointer
RT_ft
DQ
Threshold of the length of DQ = 1

Figure 4.2 The Scenario of Backing Up the Lookahead Pointer

Another way is that the DT assignment is adjusted based on the traffic load so that more cells can enter FPTQs from VCQs directly, and become easy to be accessed by the lookahead pointer. When the traffic load is not very heavy, we hope all traffic can be served as soon as possible to keep the cell transfer delay (CTD) small. In this case, the level of leaky buckets' update operation is bypassed so that a small DT can be assigned to our traffic, and let them be inserted into FPTQs. By this way, lookahead algorithm can be used in FPTQs to catch the cell with the smallest DT in the current sorting unit, and serve received cells earlier. Because the outlink is to be idle in this case, serving the cell with the smallest DT in advance will not interfere with other conforming cells. On the contrary, sending them in the idle slot can reduce the backlogged cells in the sorting unit so that the memory occupancy will be reduced. In addition, using such an algorithm, the starvation of the outlink can be avoided so that the scheduler is close to work-conserving.

When the traffic load is heavy, in order to avoid the congestion of the outlink, the scheduler switches back to leaky bucket algorithm to calculate DTs, and shapes the input traffic so that it can smooth the out stream. In this case, because of the heavy traffic load, there are always enough cells in FPTQs so that the efficiency of lookahead algorithm can be preserved.

The rate-adaptive scheduling algorithm is described as follows:

1. The process which is triggered by an arrival of a cell is shown in Fig. 4.3.

   It is almost same as the process triggered by an arriving cell in the hierarchical traffic shaper. But when the incoming cell is assigned a DT which makes the cell be appended to a FPTQ located between RT_ft and the Lookahead-pointer, it can be appended to DQ directly or pulled back the Lookahead-pointer to point to this FPTQ depending on the length of DQ larger or smaller than the threshold.

2. The process which is triggered by a departure of a cell is shown in Fig. 4.4.

   Compared to the process triggered by cell's departure in the hierarchical traffic shaper, the main difference is in the DT calculation algorithm. In the rate-adaptive scheduling algorithm, we change the DT calculation algorithm based on the traffic load. There are two cases:

   - The length of DQ > Threshold, which means the traffic load is heavy, and there are many backlogged cells in the DQ. The same DT calculation

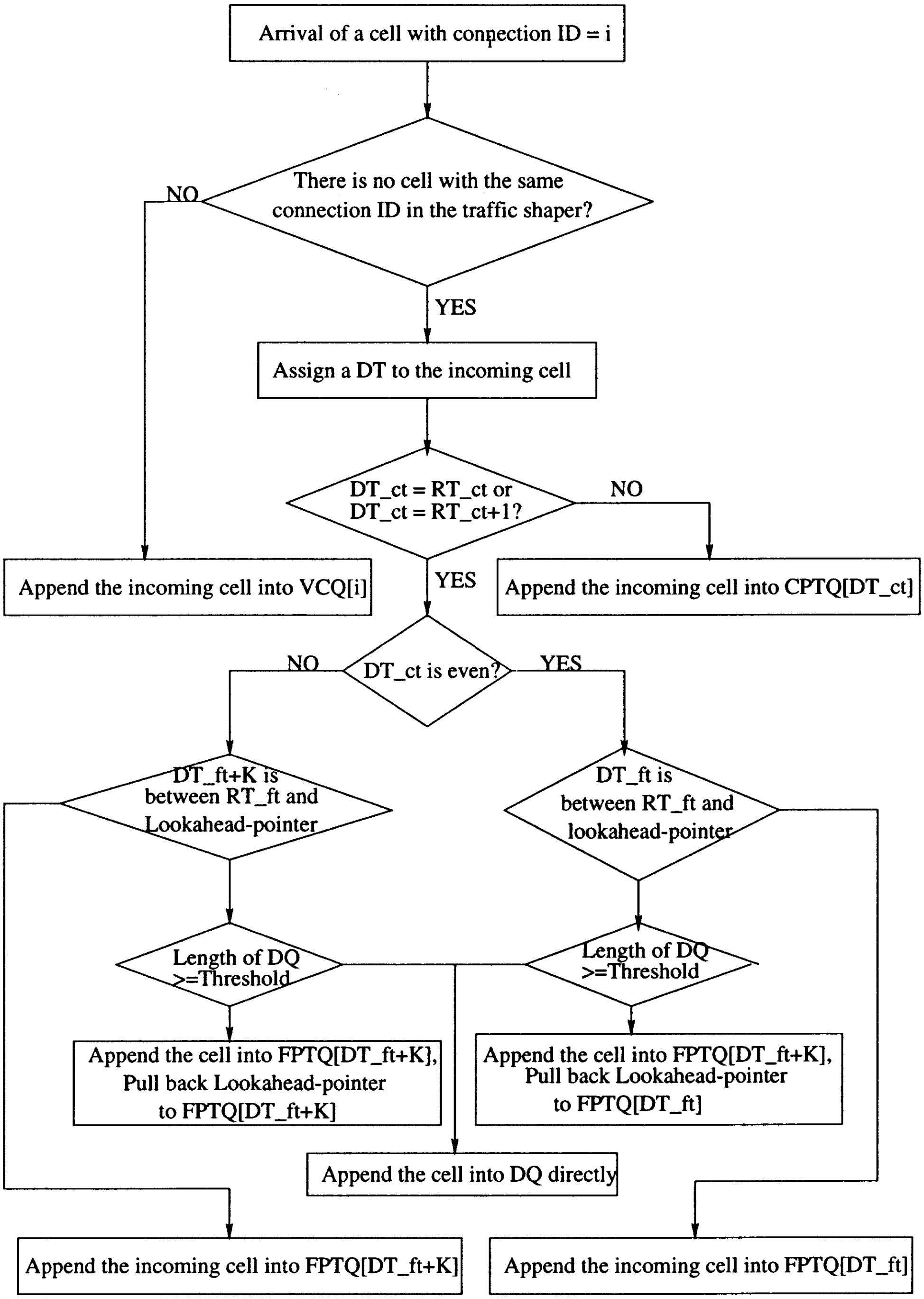

Arrival of a cell with connection ID = i
There is no cell with the same connection ID in the traffic shaper?
NO
YES
Assign a DT to the incoming cell
DT_ct = RT_ct or DT_ct = RT_ct+1?
NO
Append the incoming cell into VCQ[i]
YES
Append the incoming cell into CPTQ[DT_ct]
NO
DT_ct is even?
YES
DT_ft+K is between RT_ft and Lookahead-pointer
DT_ft is between RT_ft and lookahead-pointer
Length of DQ >=Threshold
Length of DQ >=Threshold
Append the cell into FPTQ[DT_ft+K], Pull back Lookahead-pointer to FPTQ[DT_ft+K]
Append the cell into FPTQ[DT_ft+K], Pull back Lookahead-pointer to FPTQ[DT_ft]
Append the cell into DQ directly
Append the incoming cell into FPTQ[DT_ft+K]
Append the incoming cell into FPTQ[DT_ft]

Figure 4.3 Process Triggered by the Arrival of A Cell in HRAS

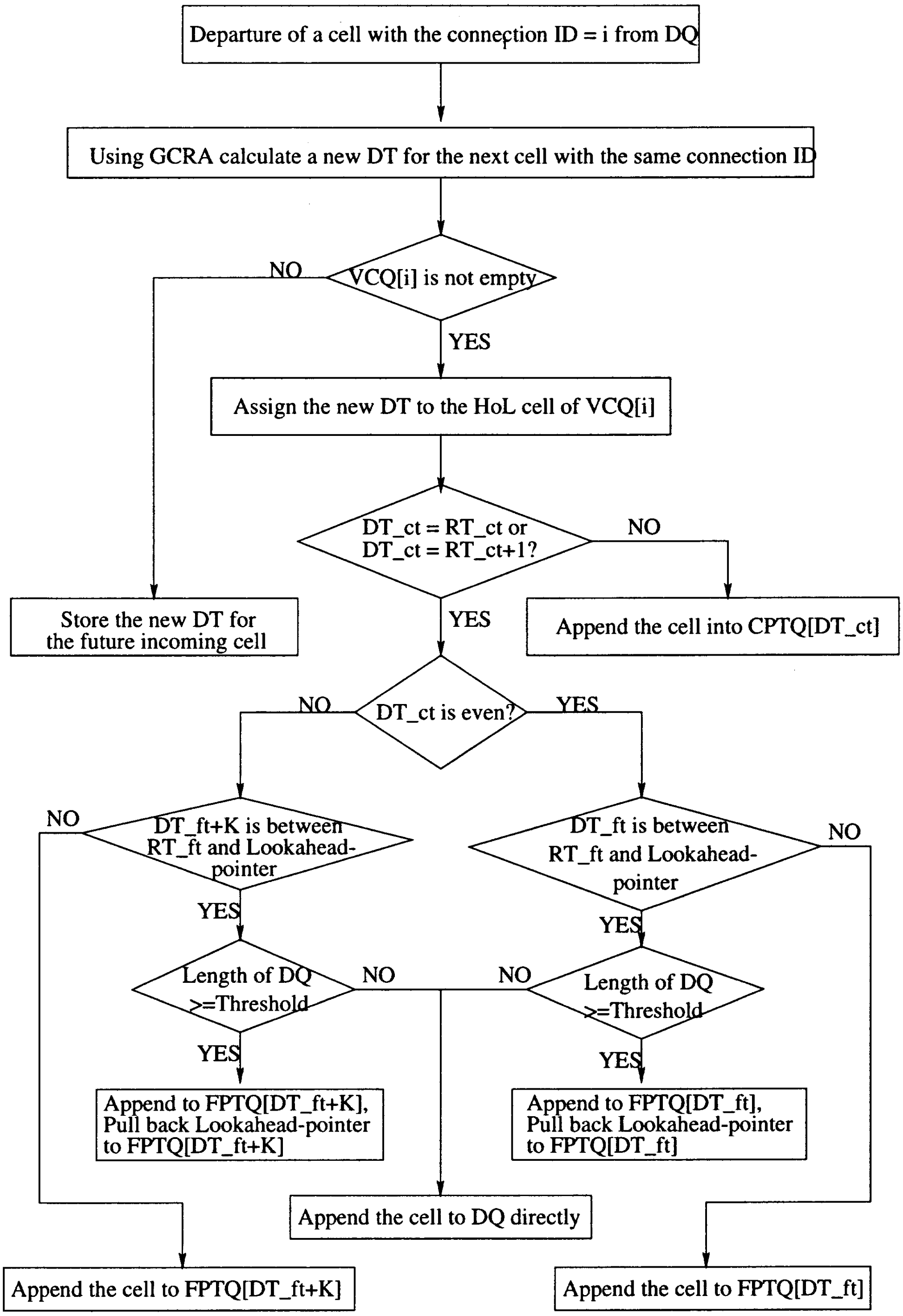


Figure 4.4 Process Triggered by the Departure of A Cell in HRAS

and bucket level update as the traffic shaper are used, which are shown in Fig. 2.8.

- The length of DQ $\leq$ Threshold, which means the traffic load is not heavy, and the outlink is to be idle, the DT assignment algorithm is adjusted:
  Step1: Update the leaky bucket by decreasing the level X by the number of the cell slots passes since the last cell's departure;
  Step2: Without increasing the level of the leaky bucket X by T_I, the next slot RT+1 is assigned as the new DT.

By this way, the sorting unit can act as a rate-adaptive scheduler by using different algorithm to assign DT. Meanwhile, it also can perform the shaping function when the traffic is heavy.

3. In each time slot, beside serving FPTQ pointed by RT_ft and moving HoL cell of CPTQ pointed by RT_ct+1 to its associated FPTQ, it is also necessary to judge whether the lookahead operation should be started: If the length of DQ < Threshold, or LookAhead-Pointer(LAP) is pointing to an empty FPTQ, then increase LAP one by one, and check each FPTQ pointed by the LAP empty or not. If the LAP finds a nonempty FPTQ, it stops moving. Otherwise, it keeps moving consecutive n FPTQs, where n is the lookahead depth limit.

4.3 Performance Evaluation of the Hierarchical Rate-Adaptive Scheduler with Lookahead Algorithm

The performances of HRAS with Hierarchical Exact Sorting Shaper, FCFS, RR, and Ideal GCRA scheduler under heavy traffic load and under light traffic load are compared in this section. Results show that it really can work like a GCRA scheduler under light traffic load, and perform shaping function when the traffic load is heavy. In both scenarios, the lookahead limit is chosen to be 5 (which is practical for current VLSI techniques), and the threshold for the length of DQ is 4. In all experiments, OC3 outlink rate is used, which is 155.52Mbps. Fig. 4.5 shows the effect of the selection of the threshold of the DQ length on the average cell transfer delay and maximum memory occupancy.

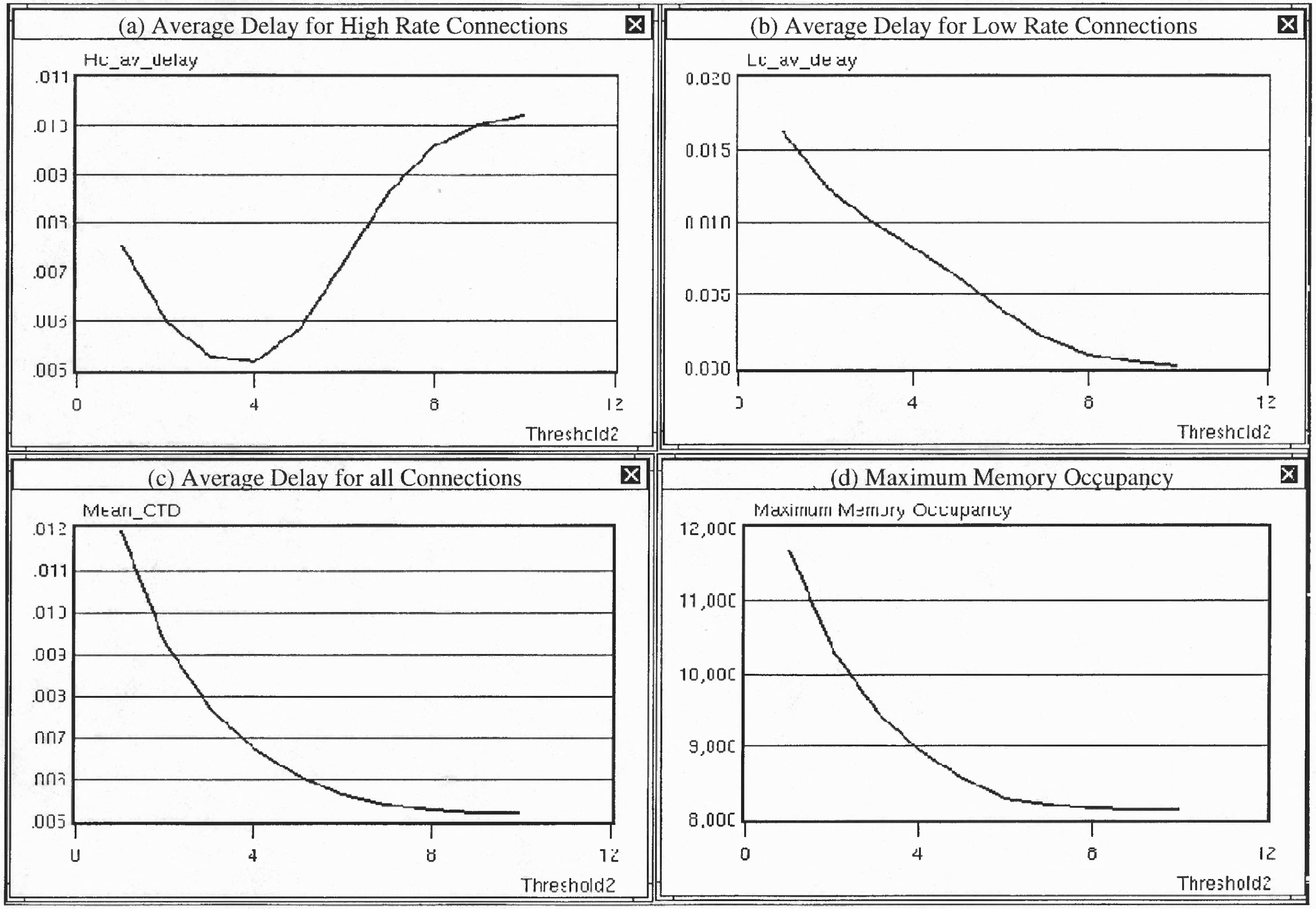
When a large threshold is chosen, the scheduler can lookahead more FPTQs to move more cells into DQ so that it can decrease the maximum memory occupancy and average delay of all connections. However, there may be more non-conforming cells from low speed connections inserted in front of cells from high speed connections when the threshold is increased over some value. Consequently, the average cell delay of high speed connections will increase. This scenario can be observed.

In heavy traffic load scenario, 4000 low speed sources and 4 high speed sources are used. All of them are VBR sources. Their source parameters are:
All low speed sources: PCR=366 cells/sec, SCR=45 cells/sec, MBS=20;
All high speed sources: PCR=366000 cells/sec, SCR=45000 cells/sec, MBS=20.

In order to observe the behavior of the rate-adaptive scheduler, it is tested under heavy traffic. When the measured traffic load $\rho = 0.9$, the performance comparison is shown in Fig. 4.6.

In light traffic load scenario, 1600 low speed sources and 4 high speed sources are used. For each source, parameters are same as previous scenario. Then, the measured traffic load $\rho = 0.6$. The performance comparison is shown in Fig. 4.7.

From these two tables, it is found that, in the heavy traffic scenario, the HRAS' cell violation ratio is much smaller than other scheduler schemes, which reflects its shaping function. Its violation ratio is just half of FCFS scheduler, and one third of RR scheduler. Meanwhile, it reduces the MMO about 30% compared to the traffic shaper, which is comparable with other work-conserving schedulers. In the light traffic scenario, HRAS' high speed connections get obvious gain in the average CTD, which is comparable with other schedulers. Although its low speed connections' CTD is larger than other scheduler schemes, it's trivial compared to the rate of low speed


(a) Average Delay for High Rate Connections

(b) Average Delay for Low Rate Connections

(c) Average Delay for all Connections

(d) Maximum Memory Occupancy

Figure 4.5 The effect of different threshold of DQ length (delay: second)

Performance item scheme	Average CTD(ms)	HC_CTD (ms)	LC_CTD (ms)	MMO	Violation Ratio
Shaper	16.6	9.72	23.5	12886	0
HRAS	6.754	5.147	8.355	8965	0.10
FCFS	5.51	5.512	5.506	8118	0.19
RR	5.473	5.176	5.773	8118	0.34
GCRA Scheduler	5.164	1.898	8.458	8118	0.12

HC_CTD: High Rate Connections' Cell Transfer Delay;
LC_CTD: Low Rate Connections' Cell Transfer Delay;
MMO: Maximum Memory Occupancy.

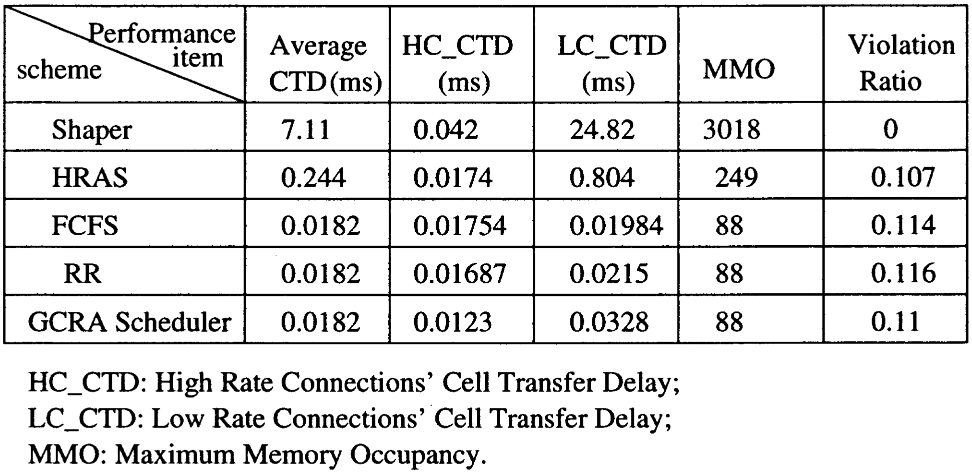
Figure 4.6 The Performance Comparison under Heavy Traffic Load $\rho = 0.9$

connection, and it is just 3% of the traffic shaper's low speed connections' CTD. This also reflects the fairness in bandwidth assignment: HRAS gives the high speed connections higher priority. In the light traffic scenario, HRAS' average CTD and MMO are smaller than the shaper, and closer to those work-conserving scheduler schemes, which reflects its scheduling function. From these results, we can say that this algorithm performs a Hierarchical Rate-Adaptive Scheduler with good fairness and traffic smoothing features.

4.4 The Hierarchical Near Work-Conserving Scheduler with Best Effort Traffic

4.4.1 The Architecture of Hierarchical near Work-Conserving Scheduler with Best Effort Traffic

Currently, with the exponential growth of the Internet traffic, there is more and more best effort traffic which has no QoS requirements. For our proposed hierarchical traffic shaper, it is very easy to reconstruct it to a hierarchical near work-conserving scheduler by using a FIFO queue to accommodate the best effort traffic. This proposed scheduler not only can guarantee QoS requirement for those services



Performance item scheme	Average CTD(ms)	HC_CTD (ms)	LC_CTD (ms)	MMO	Violation Ratio
Shaper	7.11	0.042	24.82	3018	0
HRAS	0.244	0.0174	0.804	249	0.107
FCFS	0.0182	0.01754	0.01984	88	0.114
RR	0.0182	0.01687	0.0215	88	0.116
GCRA Scheduler	0.0182	0.0123	0.0328	88	0.11

HC_CTD: High Rate Connections' Cell Transfer Delay;
LC_CTD: Low Rate Connections' Cell Transfer Delay;
MMO: Maximum Memory Occupancy.

Figure 4.7 The Performance Comparison under Light Traffic Load $\rho = 0.6$

who require QoS guarantee, but also can improve the outlink utilization to about 100 percent by introducing best effort traffic.

The hierarchical near work-conserving scheduler keeps the structure of the traffic shaper portion, and adds one Best Effort Queue(BEQ) which can be a FIFO to accommodate the best effort traffic. In the new architecture, the portion of the traffic shaper can provide QoS guarantee, and the BEQ can serve the best effort traffic. The structure is shown in Fig. 4.8.

The algorithm is:

- For the traffic with QoS requirements, this scheduler shapes it same as in the hierarchical traffic shaper.

- For the best effort traffic, once the outlink is empty, the HoL cell of the BEQ can be sent so that the hierarchical traffic shaper can be changed to a near work-conserving scheduler. When the shared memory is full, the best effort traffic is dropped at first.

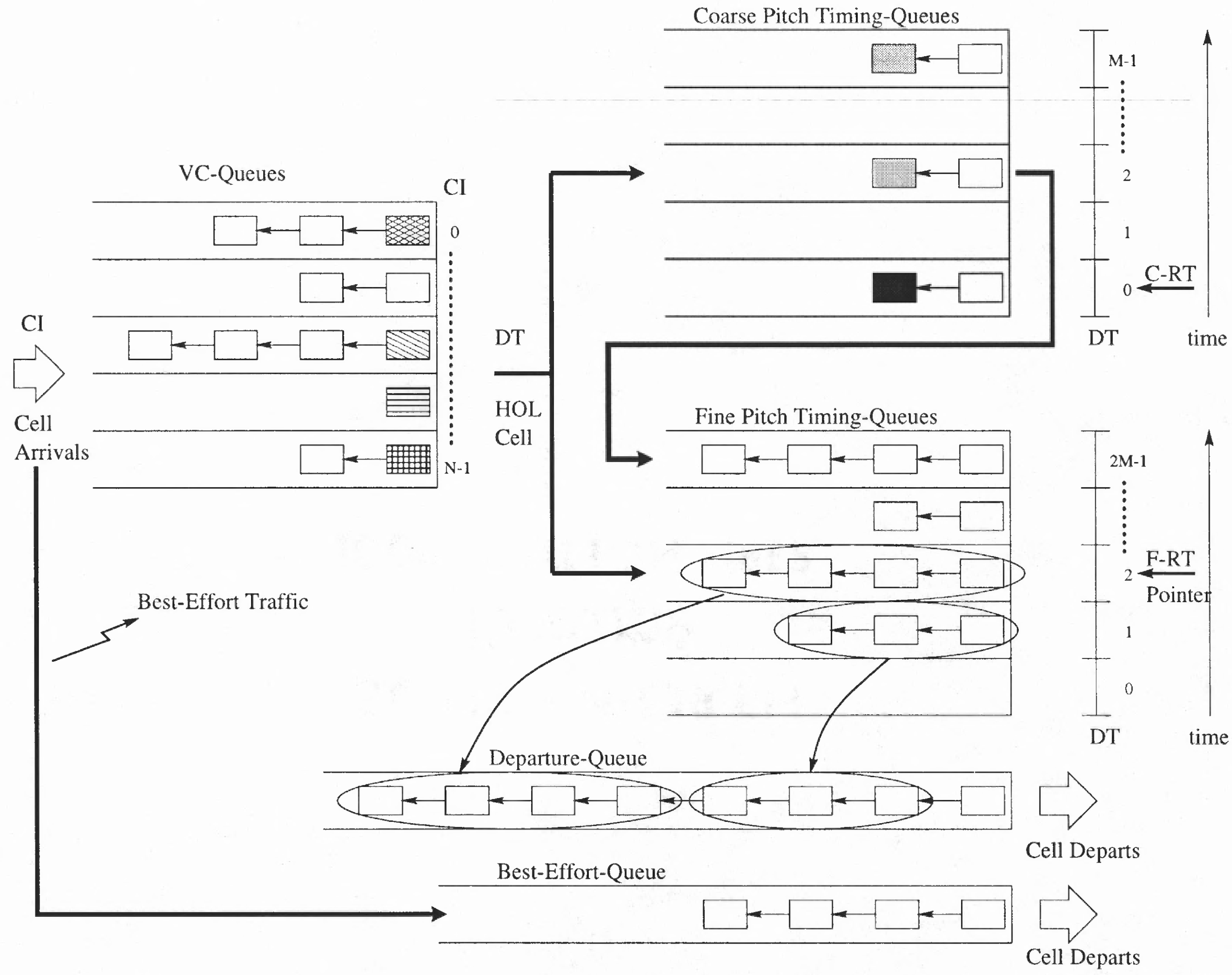


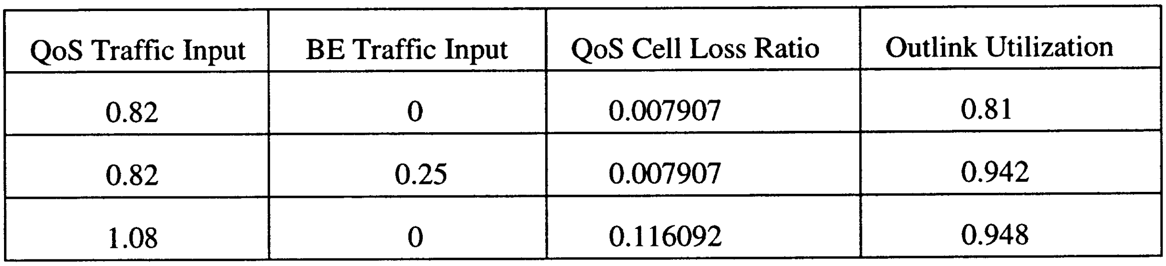
Figure 4.8 Hierarchical Work-Conserving Scheduler with BEQ

4.4.2 Performance Evaluation for the Hierarchical Work-Conserving Scheduler with Best Effort Traffic

In simulation experiments, the maximum memory occupancy is kept to be 2500 cells. From the simulation experiment, two ways to improve the utilization of the outlink are compared: one is by increasing the input traffic load of the traffic shaper, another way is by introducing best effort traffic. It can be found that the former one increases cell loss ratio much more given the same shared memory limit.

In the first experiment, about 80% of OC3 input load is applied to the hierarchical traffic shaper. In the second experiment, about 80% of OC3 input load and

25% OC3 best effort traffic is applied to the hierarchical scheduler with best effort traffic. The cell loss ratio for QoS traffic in both experiments is same, 7.907×10^{-3}, but the output link utilization in the traffic shaper is about 80% of OC3, and in the near work-conserving scheduler, it is about 94.2% of OC3. The proposed scheduler can reach 100% outlink utilization by increasing the best effort traffic load. Of course, the loss of the best effort traffic will increase. In the third experiment, in order to get 94.8% of OC3 output utilization in the traffic shaper without BEQ, the QoS traffic input load has to be increased to 1.08 of OC3. However, it leads to the 0.116 cell loss ratio for QoS traffic as observed from our simulations. It is much bigger than the scheme with BEQ. Results of these three experiments are listed in Fig. 4.9.



QoS Traffic Input	BE Traffic Input	QoS Cell Loss Ratio	Outlink Utilization
0.82	0	0.007907	0.81
0.82	0.25	0.007907	0.942
1.08	0	0.116092	0.948

(Normalized by the outlink LCR)

Figure 4.9 Performance Comparison between the Scheduler with BEQ and the Traffic Shaper

CHAPTER 5

AN IMPROVED FAIR TRAFFIC SHAPING ALGORITHM AND IMPLEMENTATION

The widely used traffic shaping and scheduling algorithm is the GCRA, which is also known as leaky bucket algorithm. This traditional GCRA proposed in ATM Forum is an Arrival Event Driven Traffic Shaping Algorithm (AEDTS). The new theoretical conforming departure time is calculated at the instance of a cell's arrival, and its calculation is based on the last theoretical conformance time. However, the contention of multiple connections at the output of the multiplexer may lead to the multiplexed out stream not to conform to their traffic contract.

In order to make the multiplexed out stream strictly conforming, a Departure Event Driven Traffic Shaping Algorithm (DEDTS) is proposed [15]. In DEDTS, the new theoretical conforming departure time is calculated at the instance of a cell's departure, and its calculation is based on the real departure time of the previous cell from the same connection which guarantees that the successive cell will strictly conform to its associated traffic contract. Nevertheless, when the traffic load is high, and there are many conforming cells waiting for the service in the Departure Queue (DQ), the cells may get high shaping delay due to losing contention at the output of the multiplexer. It is especially unfair for connections with high rate which usually require small shaping delay.

In this chapter, a new traffic shaping algorithm is proposed, which modifies the DEDTS algorithm by adding a process to resort the Completing Service Time stamps (CT) of all conforming cells. The CT is the sum of the conforming Departure Time (DT) and the service time of its associated connection. By resorting the CT stamps, the conforming cells can be prioritized based on their connection rate, and assign the available bandwidth more fairly, thus reducing the shaping delay of high rate connections. This algorithm can be easily implemented by modifying the Hierarchical Timing Queues Traffic Shaper architecture [97].

The following sections of this chapter are arranged as below. In the second section, two existing traffic shaping algorithms, AEDTS and DEDTS, are described. In the third section, the details of the proposed traffic shaping algorithm Departure Event Driven plus Completing Service Time reSorting Traffic Shaping Algorithm (DED+CTS) are described, and a hardware implementation architecture for this algorithm is given. The fourth section presents the performance evaluation of these traffic shaping algorithms, and shows that our algorithm reduce the shaping delay of high rate connections without introducing any cell violation monitored by UPC.

5.1 Revisit Existing Shaping Algorithms

Based on the GCRA, the traffic shaper assigns conforming departure time DT to each incoming cell so that it can schedule their transmission. For the assignment of departure time, two event-driven traffic shaping algorithms are proposed: Arrival event driven traffic shaping algorithm and Departure event driven traffic shaping [15, 18, 44].

- Arrival Event Driven Traffic Shaping Algorithm (AEDTS)

  In AEDTS, while a new cell is received and appended to its associated Virtual Connection Queue (VCQ), its departure time will be calculated based on the last calculated departure time of the previous cell in the same VCQ using GCRA. The leaky bucket level will be updated at the same time [18]. The algorithm is describe below:

  1. When a cell comes into the traffic shaper, it is assigned a theoretical DT, and then it is inserted into a sorting unit to wait for its DT due;

2. When a cell's DT is due, it is append to a Departure Queue (DQ) which is a FIFO queue. If it is the HoL of the DQ, it will be served immediately. Otherwise, it has to wait for its service till all conforming cells before it are served. In each cell slot, there is only one cell to be served.

The DT calculation and buckets' level update in AEDTS are shown in Fig. 2.5. For AEDTS, although each connection complies with its own leaky bucket parameters before multiplexing, as converging with other connections on the outlink, each shaped cell may violate the shaping parameter because of the contention among eligible cells for transmission [15]. If the previous cell of one VCQ is delayed due to loss of contention, the successive cell of the same VCQ may be assigned an improper departure time based on the calculated departure time of the previous cell which is earlier than its real departure time. Such improper assignment of departure time may clump the successive cell sequence so that it leads to a larger Cell Delay Variation which may violate the traffic contract CDVT and BT at the User Parameter Control (UPC). Another shortcoming of AEDTS is that, in each cell slot, there may be more than one cell coming, then, multiple conforming departure times have to be calculated in one cell slot. In this case, it may become a bottleneck when the traffic load is high. It also requires a large memory for storing such a large number of time stamps.

- Departure Event Driven Traffic Shaping Algorithm (DEDTS)

In order to strictly limit the CDVT and BT at the output of the shaping multiplexer [15, 44], Departure Event Driven Traffic Shaping Algorithm (DEDTS) was proposed. In DEDTS, the new departure time DT is calculated based on the real departure time of the previous cell while its departure, and the leaky bucket level is updated at the same time. The algorithm is described below:

1. A cell arrives: if there is no other cell from the same connection in the traffic shaper, the new coming cell is assigned a DT, and insert into the sorting unit to wait for its DT due. Otherwise, it will be appended to its associated VCQ;

2. A cell leaves: the new theoretical DT is calculated based on the last real DT. If the leaving cell's associated VCQ is not empty, then the HoL of this VCQ will be assigned the new calculated DT, and insert to the sorting unit to wait for its DT due. Otherwise, the new calculated DT will be stored for the next cell from the same connection;

3. A cell's DT due: it will be appended to the DQ. If the DQ is empty, it will be served immediately. Otherwise, it has to wait for the service till all cells before it are served.

The DT calculation and buckets' level update in DEDTS are shown in Fig. 2.8. Because the DT is calculated based on the real departure time of the previous cell, DEDTS can guarantee strict conformance to CDVT value of each connection. Meanwhile, in each time slot, only one cell is transmitted, so only one new conformance time DT is calculated, which also eliminates the bottleneck due to multiple DT calculation for more than one cell arriving at the same cell slot in AEDTS. However, when the traffic load is heavy, there may be many conforming cells in DQ waiting for service. Cells losing contention have to wait some extra time for transmission. It is shown that the extra delay may be accumulated for high rate connections which can lead to high shaping delay. This is an undesired result, that may lead to unfair treatment especially for the high rate connections.

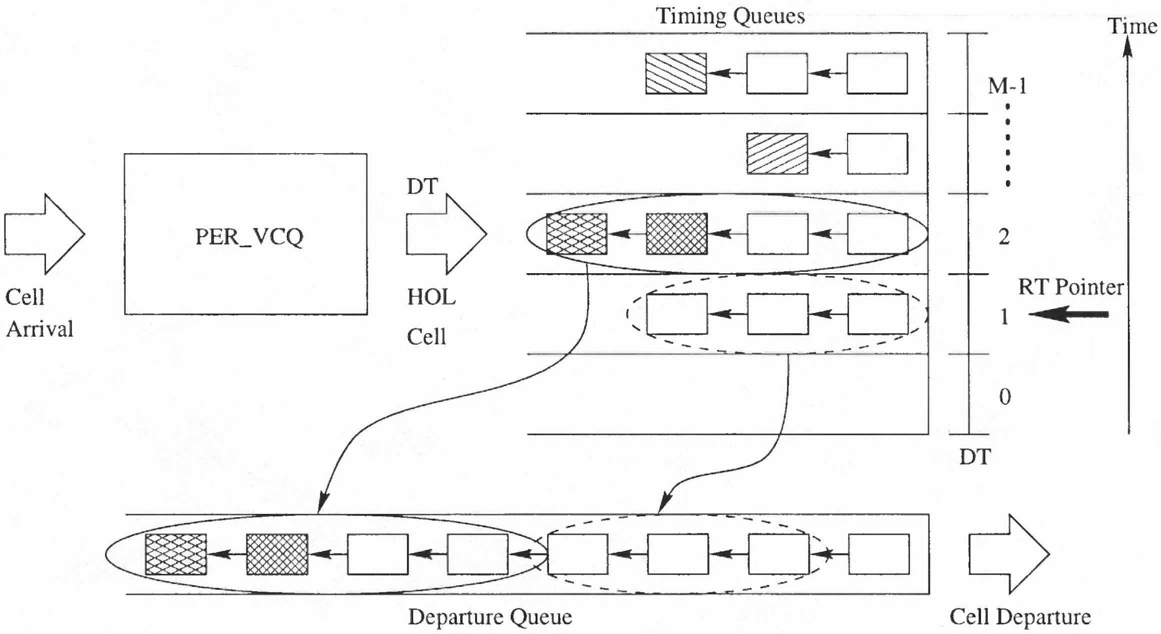


Figure 5.1 Functional Model of Conventional DEDTS algorithm

5.2 Departure Event Driven plus Completing Service Time ReSorting (DED+CTS) Traffic Shaping Algorithm

In order to avoid the extra shaping delay accumulation for high rate connections while keeping zero cell violation monitored by UPC, we propose a new traffic shaping algorithm: Departure Event Driven plus Completing Service Time Resorting (DED+CTS) Traffic Shaping Algorithm.

5.2.1 Motivation and Description of the Algorithm

Fig. 5.1 shows the functional model of conventional DEDTS algorithm.

Let us assume the current time is RT, all cells with DT equal to RT will be appended DQ. In the DQ, the order of cells is based on FIFO discipline. In each cell slot, only one cell can be sent when the DQ is not empty. When the traffic load is heavy, the DQ may be very long which leads to a large shaping delay. For high rate connections, when the length of DQ is larger than the buckets' increment, which is equal to the reciprocal of connection's rate, their leaky buckets will become ineffective. The reason is that the extra shaping delay in DQ makes cells from same connection depart far away from each other, and the level of buckets always keeps around zero. In this scenario, due to the extra shaping delay in the DQ, the high rate connection has to reduce its output rate passively, and can not get a fair bandwidth assignment. In addition, because the increment of a high rate connection's buckets is much smaller than that of a low rate connection's, a high rate connection has much larger chance to suffer when the length of DQ is larger than its bucket' level increment. This leads to an unfair bandwidth assignment. An example of this problem is shown in Fig. 5.2.

When the total amount of bandwidth is set, the number of high rate connections is much smaller than the number of low rate connections. For example, when the bandwidth is 2Mbps, there can be 10 connections with the rate 10^5bps, and 1000 connections with the rate 1000bps, and their increments I's are 20 and 2000, respectively. The high rate is 100 times of the low rate. In this scenario, if there are only high rate connections, the length of DQ can not be larger than their increment 20 because the maximum DQ length is only the number of connections 10. However, when there are another 1000 low rate connections multiplexed with these 10 high rate connections, the maximum DQ length can be more than 1000. It means that cells from high rate connections may get 1000 slot time extra queueing delay from the DQ. In this case, the DQ length will affect the performance of high rate connections seriously, but the low rate connections will not be affected obviously because the length of the DQ is definitely smaller than their increment 2000. This example shows that the conflict is between the high rate connection group and the low rate connection group, not among the high rate connections or low rate connections themselves. In Appendix I, the analysis for the length of the DQ will be given. It has higher probability that the length of the DQ will reach and be beyond the incremental intervals of connections as the load becoming heavier. In order to resolve

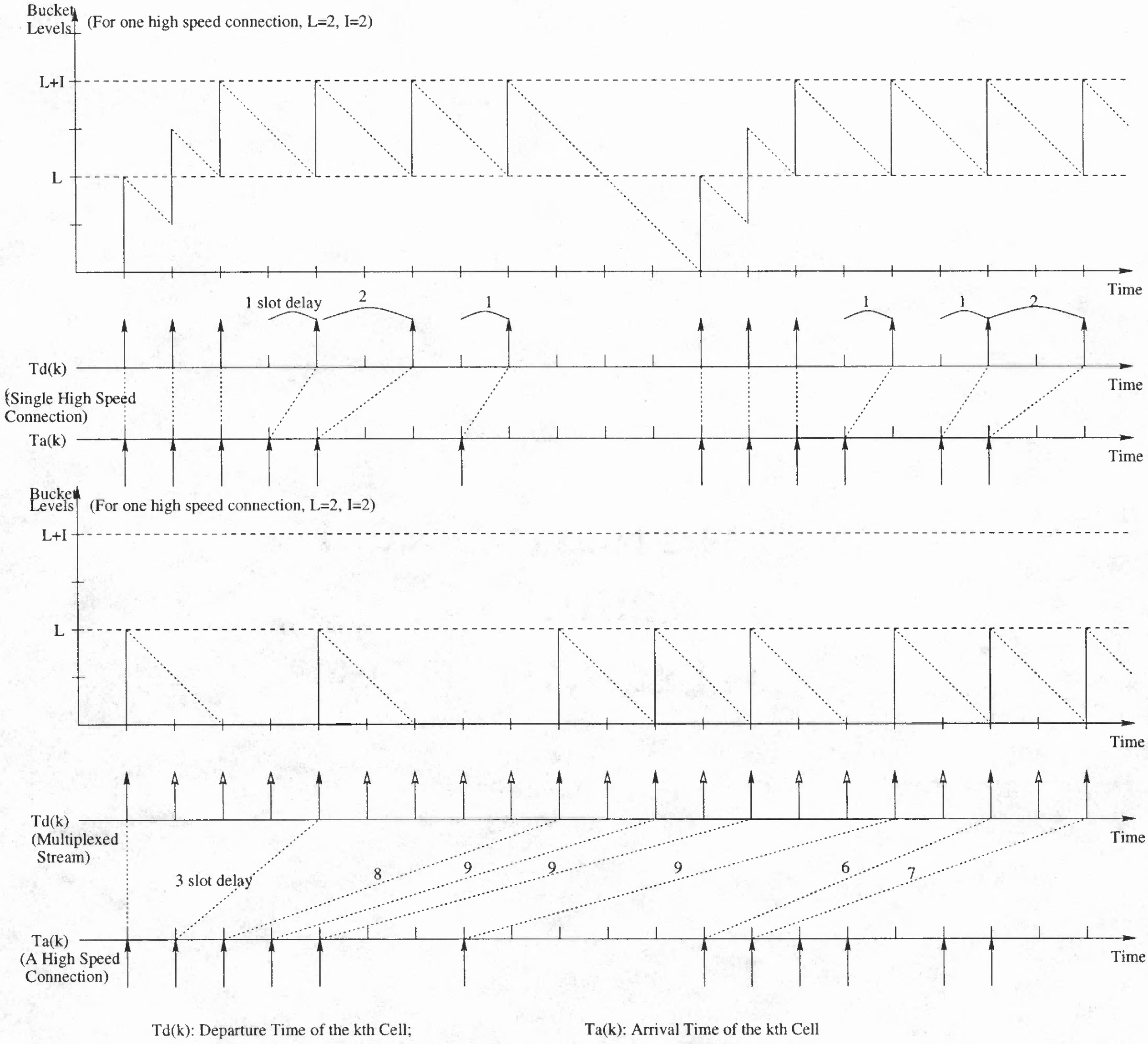

Bucket Levels
(For one high speed connection, L=2, I=2)
L+I
L
Time
1 slot delay
2
1
1
1
2
Td(k)
(Single High Speed Connection)
Ta(k)
Time
Time
Bucket Levels
(For one high speed connection, L=2, I=2)
L+I
L
Time
Td(k)
(Multiplexed Stream)
3 slot delay
8
9
9
9
6
7
Ta(k)
(A High Speed Connection)
Time
Time
Td(k): Departure Time of the kth Cell;
Ta(k): Arrival Time of the kth Cell

Figure 5.2 Extra Shaping Delay in DEDTS (L=2, I=2, GCRA(2,2))

the conflict, and let the high rate connection get its deserved bandwidth, an improved fair traffic shaping algorithm is proposed in this chapter.

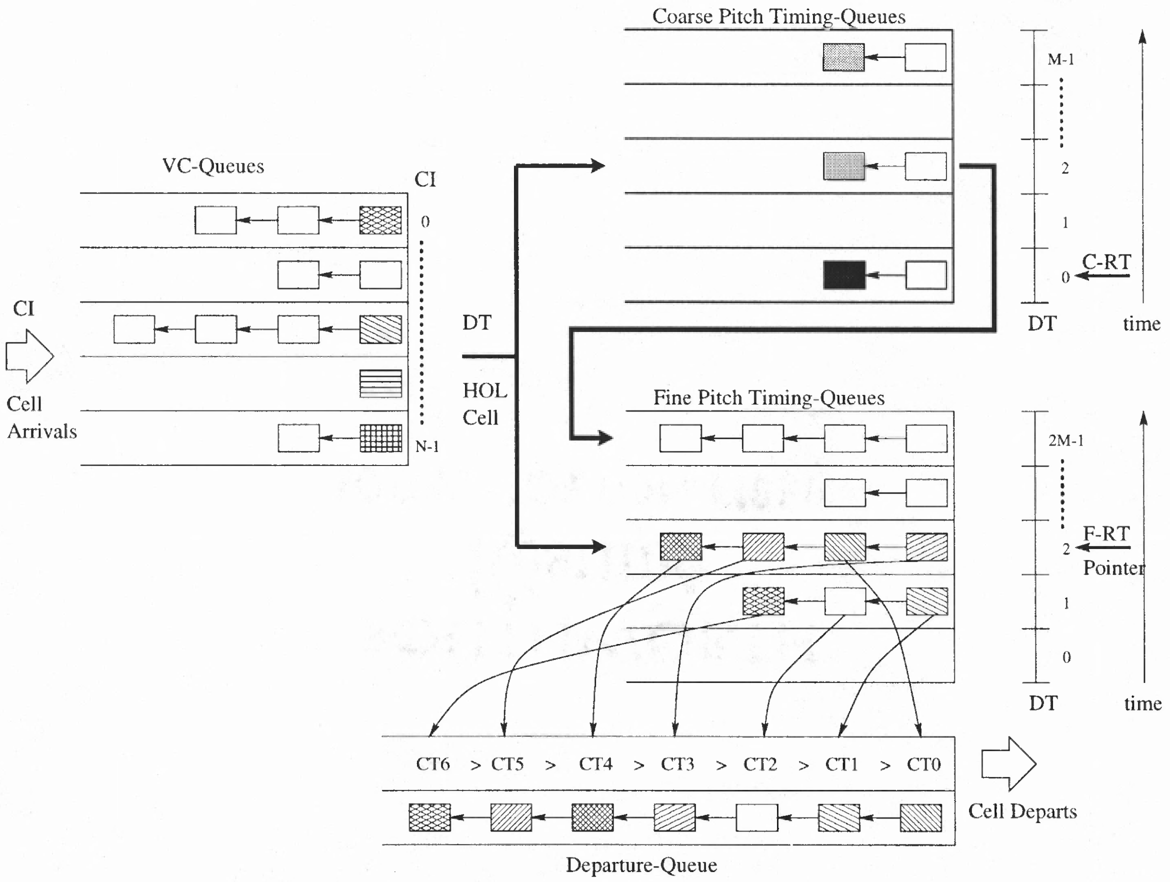
In the proposed algorithm, the FIFO DQ is replaced with a sorting DQ which resorts all conforming cells in DQ based on their Completing Service Time(CT). The CT is equal to the cell's DT plus its associated connection's bucket level increment T_I which is the reciprocal of the connection's assigned bandwidth, which is its SCR, $CT = DT + T_I$, and $T_I = 1/SCR$. The increment T_I also represents the service time of a connection. Because only conforming cells are resorted, there is no cell violation introduced by this operation. The underlying principle of resorting on CT is to prioritize all conforming cells based on their rates: high rate connections can get higher priority so that their shaping delay can be reduced accordingly, and low rate connections get lower priority and higher extra shaping delay. Because CTs include the rate information, such a prioritization process can reflect better fairness in bandwidth assignment, and avoid the rigid prioritization which may favor some groups much more than others unfairly.

5.2.2 Implementation of the Algorithm

For the implementation of the algorithm, the Hierarchical Timing Queues Traffic Shaper architecture (HTQTS) proposed in [95] can be modified by replacing the FIFO DQ with a sequencer [18] DQ. Because only conforming cells are resorted in DQ, the sorting capability requirement is not very high, the sequencer series can be used to implement the resorting. The architecture is shown in Fig. 5.3. The sequencer is a sorting queue. It can sort cells based on their CTs.

The overall operation process is described below:

1. At the instance of a cell's arrival, if there is no other cell from the same connection in the traffic shaper, the new coming cell is assigned a DT, and insert



DT: Departure Time;
CT: Complete Service Time

C-RT: Coarse Pitch Part of Real Time;
F-RT: Fine Pitch Part of Real Time

Figure 5.3 The Implementation Architecture of DED+CTS Algorithm

into the sorting unit to wait for its DT due. Otherwise, it will be appended to its associated VCQ;

2. When a cell leaves, the new theoretical DT is calculated based on the last real DT. If the leaving cell's associated VCQ is not empty, then the HoL of this VCQ will be assigned the new calculated DT, and inserted to the sorting unit to wait for its DT due. Otherwise, the new calculated DT will be stored for the next cell from the same connection;

3. When a cell's DT is due: its DT is updated to CT by adding its service time $T_I = 1/SCR$ to its DT. Then the cell is inserted to the sequencer DQ based on its CT.

4. In each cell slot, if the DQ is empty, the HoL cell will be served.

5.3 Performance Evaluation for DED+CTS Algorithm

In this section, the performance of DED+CTS algorithm and DEDTS algorithm is compared, in which, DEDTS algorithm is implemented by the HTQTS with a FIFO DQ at the output of the shaper. The performance evaluation includes two scenarios via simulations: a heavy traffic load scenario and a light traffic load scenario. In the experiments, 4040 VBR sources are used. In the heavy traffic load scenario, 40 of them are high speed sources with the PCR 10000 cells/sec, and SCR 7500 cells/sec; 4000 of them are low speed sources with the PCR 100 cells/sec, and SCR 75 cells/sec. Meanwhile, the measured traffic load in the simulation is $\rho = 0.95$. In the light traffic load scenario, the high speed sources are with the PCR 6000 cells/sec, and SCR 3800 cells/sec; the low speed sources are with the PCR 70 cells/sec, and SCR 40 cells/sec.

In an ideal shaping multiplexer,

$$\overline{T_{si}} \propto T_i, \tag{5.1}$$

T_i is the increasement of the leaky bucket i, and the $\overline{T_{si}}$ is the average shaping delay of connection i. T_i represents the bandwidth requirement of connection i, and T_{si} reflects the bandwidth occupancy of the connection i, based on the fairness principle of GPS [69], this proposition stands. In the Appendix II, the GPS is summarized, and it shows that the proposed algorithm is a special implementation case of the GPS, and it inherits the fair scheduling feature of GPS.

Based on the discussion above, it can be got that in a fair shaping multiplexer,

$$\frac{\overline{T_{si}}}{T_i} = \frac{\overline{T_{sj}}}{T_j}, \tag{5.2}$$

for all i, j, and connection i, j are active connections in the shaping multiplexer.

For comparison purposes, let us define the fairness coefficient

$$FC = \frac{\overline{T_{si}}}{T_i} = R_i \times \overline{T_{si}}, \tag{5.3}$$

in which, R_i is the committed rate of this connection, and the increment of the leaky bucket for this connection $T_i = 1/R_i$, as we discussed in the previous section. When the FCs for all connections are equal, the network resources allocation gets to the most fair point. Therefore, the fairness of the resources allocation between any two connections i and j can be defined as the ratio: $F = FC_i/FC_j$ (F=1 corresponds to the optimal fairness). The emphasis of this performance analysis is placed on the comparison of the fairness characteristics of the two traffic shaping algorithms.

The corresponding results are presented in Fig. 5.4

			SCR (Cells/sec)	Average Delay (seconds)	F (FCl/FCh)	Violation
Heavy Load (Load= 0.95)	DEDTS	High Speed Connection Group	7500	7.60e-03	0.02	0
		Low Speed Connection Group	75	1.16e-02		
	DED+CTS	High Speed Connection Group	7500	1.92e-04	0.99	0
		Low Speed Connection Group	75	1.91e-02		
Light Load (Load= 0.55)	DEDTS	High Speed Connection Group	3800	1.06e-04	1.04	0
		Low Speed Connection Group	40	1.05e-02		
	DED+CTS	High Speed Connection Group	3800	1.05e-04	1.03	0
		Low Speed Connection Group	40	1.05e-02		

Figure 5.4 The Performance Comparison for DED+CTS and DEDTS

From this figure, it is observed that the performance of DEDTS and DED+CTS are similar in the light traffic load scenario. However, in the heavy traffic load scenario, the delay of the high speed connections in the DEDTS algorithm increases significantly (about 4 times larger than in the DED+CTS algorithm). In DED+CTS algorithm, the delay of the high speed connections are decreased at the price of slight increase of the delay of the low speed connections. The value of F in the DED+CTS is 0.99, which is much closer to 1 than in the DEDTS algorithm. Observing the value of F for the two algorithms under the heavy load scenario, it can be concluded that the DED+CTS algorithm can achieve almost the optimal fairness (i.e. F=0.99), while the corresponding value of the DEDTS (i.e. F=0.02) indicates that the DEDTS algorithm results in a very unfair resource assignment. Meanwhile, the DED+CTS doesn't introduce any violation of CDVT and BT in the UPC.

In the proposed algorithm, by adding the completing service time stamp resorting for those conforming cells, higher priorities can be naturally given to those high speed connections, while their conformance can be guaranteed. Consequently, the serious performance degradation of the high speed connections in the heavy traffic load can be avoided without increasing total available bandwidth. The performance improvement of those high speed connections only introduces slight performance degradation of those low speed connections. The simulations show that this proposed algorithm DED+CTS can give a better fairness performance without introducing any violation at the UPC. In addition, it can be implemented easily by replacing the FIFO DQ of the HTQTS [95] with a small sorting unit which can be a sequencer [18].

CHAPTER 6

A DUAL LEVEL ACCESS CONTROL ARCHITECTURE FOR DIFFSERV NETWORKS

6.1 Motivation for Dual Level Access Control Architecture for DiffServ Networks

Currently, the Internet can not support any Quality of Service (QoS) requirements. With the growth of the Internet, more and more applications demand QoS support from the networks. To meet these demands, Internet Engineering Task Force (IETF) developed the Integrated Services architecture and its associated signaling protocol Resource Reservation Setup Protocol (RSVP). Intserv assumes that networks can guarantee service requirements to each traffic flow in an isolated way [4]. It turned out that such a flow-based architecture is unscalable because there can be thousands, even ten thousands of flows in the Internet at the same time. In order to resolve the scalability problem, IETF propose the Differentiated Services (DiffServ) architecture [4, 91]. In DiffServ networks, all flows are merged to a small number of classes based on their QoS requirements and characteristics. All traffic control mechanisms are deployed to guarantee QoS to the traffic class level so that the architecture can scale for the Internet.

In the DiffServ networks, the traffic conditioner is one of the most important parts. It includes meter, marker, shaper, and dropper. A traffic stream is classified by a classifier, then forwarded to a traffic conditioner. The meter measures the traffic stream against a traffic profile. The state of the meter may trigger a marking, dropping or shaping action [5]. The logic architecture of a packet classifier and traffic conditioner is shown in Fig. 6.1.

In the IETF draft, some policing schemes have been proposed [41, 42]. Recently, a shaper scheme for the DiffServ networks is proposed in IETF RFC [6, 26]. However, it doesn't resolve the problem of the flows fair sharing of the resources in the same class. In this chapter, we propose a Dual Level Leaky Bucket Traffic Shaper

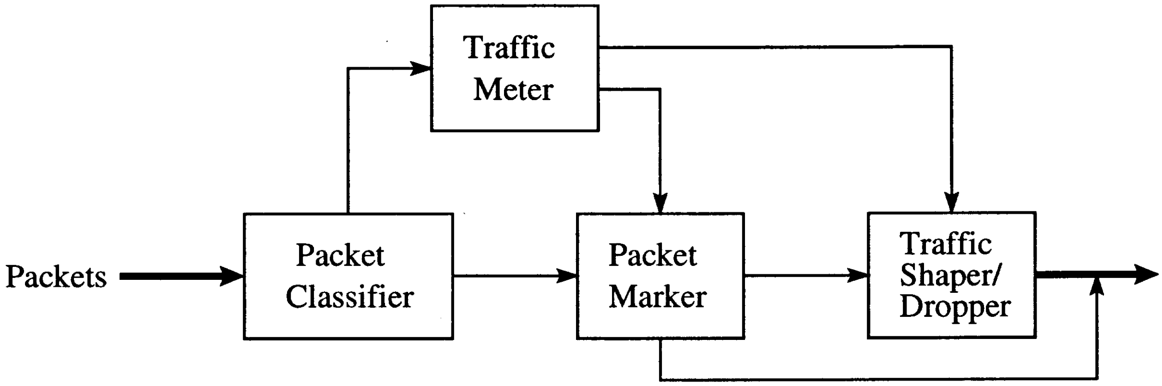


Figure 6.1 Logic Architecture of a Packet Classifier and Traffic Conditioner [5]

(DLLBTS) architecture for the edge nodes of Diffserv networks. In DLLBTS architecture, there are two levels of leaky buckets. The first level is a set of flow-based leaky buckets. It can shape flows based on the traffic profile of each flow so that it can guarantee that all flows can fairly share the network resources in the same class when they merge into one class. The second level is a set of class-based leaky buckets same as all other traffic shapers for interior nodes of the DiffServ networks. They can guarantee the aggregate behavior of a class conforming to the Service Level Agreement (SLA). However, two sets of leaky bucket traffic shaper need two sets of sorting units, and generally, the sorting unit dominates the complexity of the implementation of a traffic shaper. It is impractical to implement two sets of complex sorting units in a traffic shaper unit. We propose a simplified DLLBTS which combines the two levels of leaky buckets to one stage so that only one set of sorting unit is needed. From the simulation results, we show that the simplified DLLBTS can reach the goal of DLLBTS while keeping the low implementation complexity [96].

The following sections of the chapter are arranged as follows. The second section proposes the DLLBTS architecture, and gives the packet version leaky bucket algorithm. The third section describes the simplified DLLBTS architecture. The fourth section gives the performance analysis. The last section draws the conclusion.

6.2 Dual Level Leaky Bucket Traffic Shaper for DS Networks

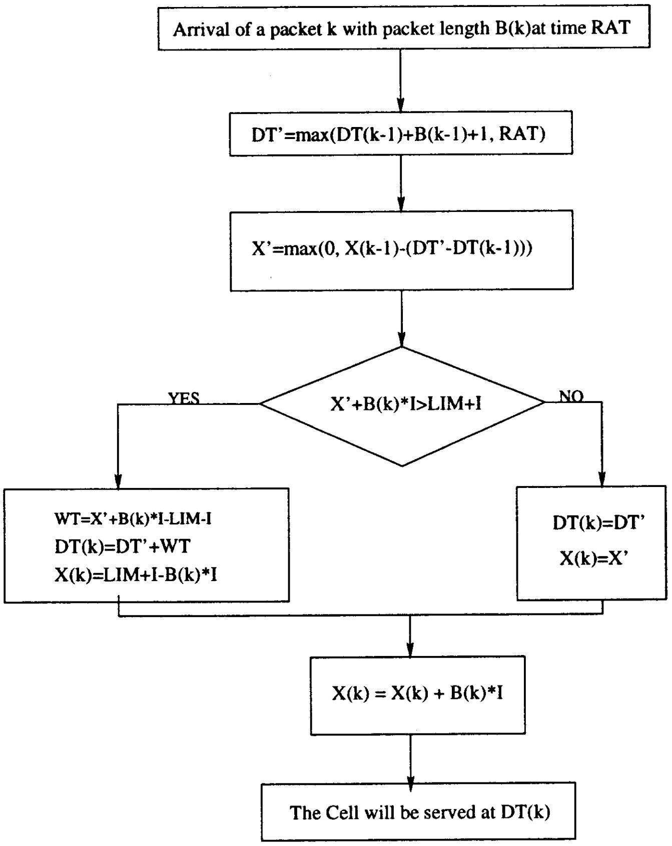
6.2.1 Leaky Bucket Traffic Shaping Algorithm for Variable Length Packet

Leaky Bucket Traffic Shaping Algorithm is generally used in ATM networks to shape the traffic based on the traffic contract which was constructed when the connection was accepted by the network. Now, we extend it into DiffServ networks to regulate the accepted traffic. However, in the DiffServ Networks, the traffic consists of packets with variable length. The leaky bucket algorithm has to be revised to work in the DiffServ networks.

In this dissertation, only the discrete-time leaky bucket algorithm is discussed. It means that the time is discretized into small operation time units which are called time slots. In ATM networks, the time is discretized into the cell slot time units. The cell slot time is the reciprocal of the Line Cell Rate (LCR). This section briefly outlines the basic changes required for the operation of the Packet-version Leaky Bucket algorithm. For a detailed description of the conventional Leaky Bucket Algorithm, the interested reader is referred to [44]. In the Packet-version Leaky Bucket (P-LB) algorithm, the time is discretized into the byte slot time units. The byte slot time unit is the reciprocal of the Line ByTe Rate (LBTR). Assuming that the LBTR is R_o bytes/sec, the negotiated rate of the flow is R bytes/sec, and the packet size is B bytes, then the increment of the leaky bucket $I = R_o/R$, and the P-LB algorithm is shown in Fig. 6.2.

6.2.2 Dual Level Leaky Bucket Traffic Shaper

In the DiffServ network frame, all proposed traffic regulations are class-based, including the traffic shaping mechanism. However, class-based traffic shaping mechanism only can control the aggregate behavior, and guarantee the aggregate traffic conforming to the Service Level Agreement (SLA) which is the negotiated traffic profile constructed when the traffic is accepted by the network. Generally,


Arrival of a packet k with packet length B(k)at time RAT
DT'=max(DT(k-1)+B(k-1)+1, RAT)
X'=max(0, X(k-1)-(DT'-DT(k-1)))
X'+B(k)*I>LIM+I
YES
NO
WT=X'+B(k)*I-LIM-I
DT(k)=DT'+WT
X(k)=LIM+I-B(k)*I
DT(k)=DT'
X(k)=X'
X(k) = X(k) + B(k)*I
The Cell will be served at DT(k)

RAT: Real Arrival Time; DT: Departure Time; DT', X': Auxiliary Variable
X: Level of the Leaky Bucket; LIM: Limit of the Leaky Bucket;
T: Incremental interval of the Leaky Bucket; WT: Waiting Time

Figure 6.2 Packet-Version Leaky Bucket Algorithm

there are more than one traffic flow in a class. All flows in the same class have to share the limited network resources assigned to this class by the network operator. If no traffic control mechanism is employed to regulate these flows, simply using First In First Out (FIFO) scheme to merge all flows into one class may lead to some unfair network resources occupancy by some malicious flows.

In order to guarantee that all flows share the resources in a class fairly, we proposed a Dual Level Leaky Bucket Traffic Shaper (DLLBTS) architecture for the edge nodes of DiffServ networks. The architecture is shown in Fig. 6.3.

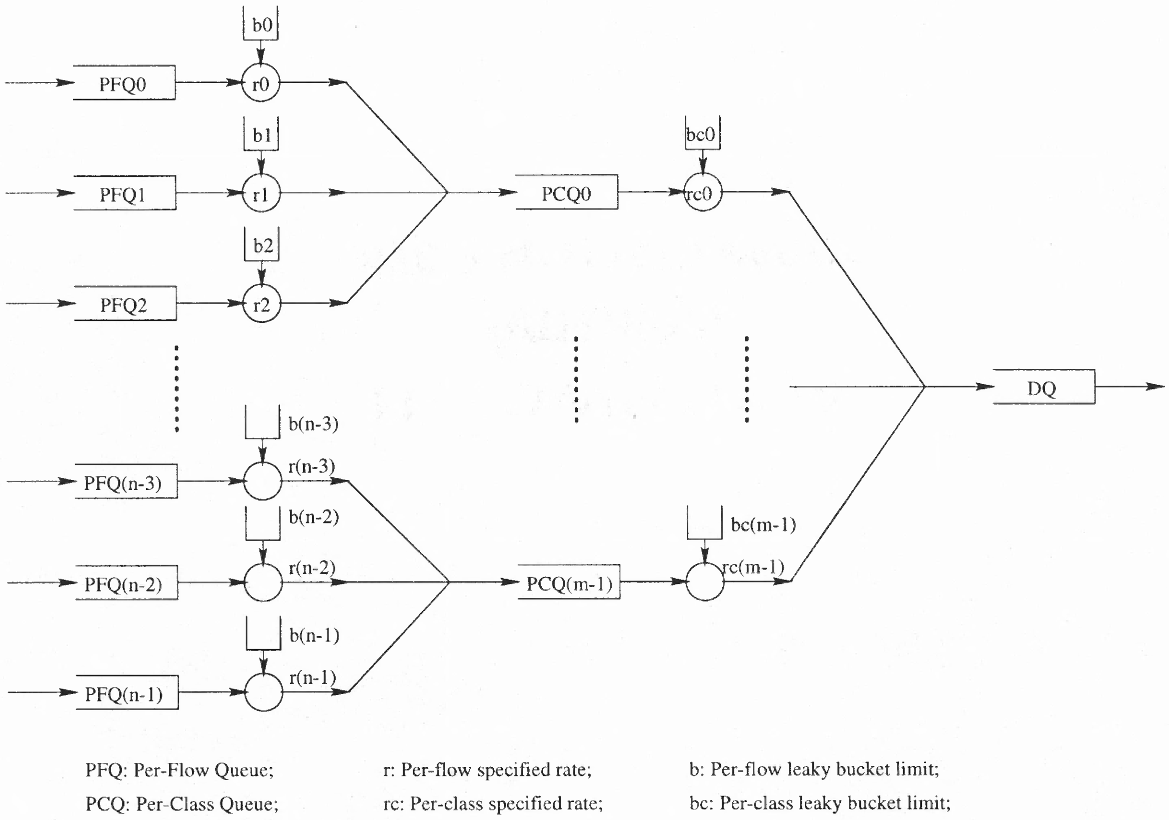


Figure 6.3 Dual Level Traffic Shaper Architecture in the Edge Node of DiffServ Network

In this figure, the left part is the local access network, the right part is the DiffServ backbone network, and at the input port of the edge node, there are a set of per-flow queues (PFQs) for all flows coming from the local access network. The first level leaky buckets are a set of flow-based token buckets. It controls the bandwidth occupancy of each flow based on the traffic contract constructed at the instance of the admission of the flow. In this figure, b_i represents the bucket limit or burst size of the flow, and r_i represents the committed rate of the flow. Therefore, all flows can share the resources in one class fairly. For each packet, a flow-based conforming Departure Time (DT) is calculated by its associated flow-based leaky bucket on its arrival. All packets are sorted by the first level leaky bucket traffic shaper, and all sorted packets of the same class are inserted into their commonly associated Per-Class Queue (PCQ). In the PCQ, a class-based DT is calculated for each packet. The second level leaky buckets are a set of class-based ones which are used to restrain the aggregate behavior of a class. Their parameters are bc_i and rc_i which represents the bucket size and the committed rate for each class respectively. They sort these class-based DTs. When these DTs are due, their associated packets are sent to Departure Queue (DQ) to wait for serving [98].

6.3 Practical Implementation DLLBTS Architecture

The DLLBTS architecture described in the previous section requires two sets of sorting units to sort these two sets of DTs. However, the implementation complexity and cost of a shaper are dominated by the complexity of sorting time stamps [95]. Two sets of sorting units increase the implementation complexity of DLLBTS. In the following we refer to this architecture as *Ideal_DLLBTS* (*I_DLLBTS*). In order to reduce the implementation complexity while implementing both flow-based and class-based control on the traffic, we propose a simplified practical implementation DLLBTS architecture (referred to as *S_DLLBTS*), as shown in Fig. 6.4.

In this architecture, there is only one set of time-stamp sorting unit which reduce the implementation complexity. At the input of the architecture, there are a set of Per-Flow-Queues (PFQ). At the output of the sorting unit, there are a set of Per-Class-Queues (PCQ) and a Departure Queue (DQ). The sorting unit can use the hierarchical timing queue architecture which is proposed in [95]. The operation processes of this architecture are explained as the following.

1. The operation process triggered by the arrival of a packet is shown in Fig. 6.5.

   When a packet with Flow ID (FID) i and Class ID (CID) j is received by the shaper, if its Per-Flow-Queue i (PFQ(i)) is not empty or the flow-based DT (FDT) of the previous packet of the same flow is not due, then it is appended to PFQ(i) to wait for its previous packet of the same flow due. If its PFQ(i)

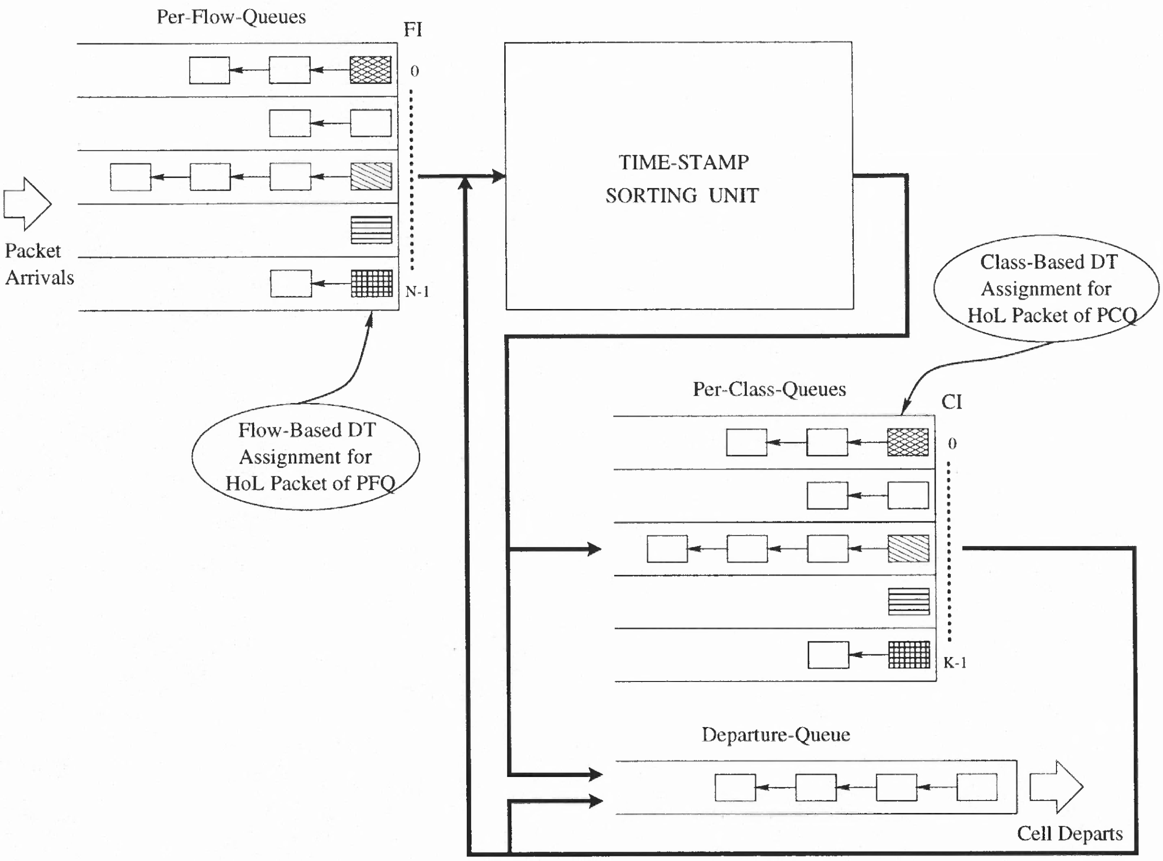


Figure 6.4 Simplified DLLBTS Architecture in the Edge Node of DiffServ Network

is empty, and the FDT of the previous packet of the same flow has been due, then a new FDT is calculated, and assigned to this new packet. This packet is inserted into the sorting unit to be sorted and stored till its FDT is due. Then it is output from the sorting unit, and sent to Per-Class-Queue (PCQ(j)). If its PCQ(j) is not empty or the new class-based DT (CDT) is not ready, it is appended into PCQ(j) to wait for its CDT assignment. If its PCQ(j) is empty, and a new CDT is ready, then the pre-calculated CDT is assigned to this packet. If this CDT is equal to or smaller than the current time, then this CDT is due or overdue, and this packet should be appended into DQ directly. Otherwise, it should be insert back into the sorting unit for resorting based on its CDT, and stored in the sorting unit till its CDT is due. When its CDT is due, it is output from the sorting unit again, and sent to the DQ to wait for serving.

2. The operation process triggered by the departure of a packet is shown in Fig. 6.6.

   When a packet with FID i and CID j departs from the DQ, a new CDT of this class is calculated. If the PCQ(j) is empty, then the CDT is stored to wait for the next packet of the same class coming. Otherwise, this CDT is assigned to the HoL packet of PCQ(j). Then this packet is inserted back to the sorting unit till its CDT is due.

In the packet format, there will be a field to differentiate the packets which are inserted to the sorting unit from the PFQ, and which are from the PCQ. By this way, two sets of DTs are sorted in a single set of sorting unit successfully, and DLLBTS is realized.

6.4 Performance Evaluation for Dual Level Leaky Bucket Traffic Shaper Architecture

In this section, the performances of our proposed architectures and mechanisms (both the *I_DLLBTS* and *S_DLLBTS*) with the corresponding performances of a single level Class-Based Leaky Bucket Traffic Shaper (CBLBTS) and a single level Flow-Based Leaky Bucket Traffic Shaper (FBLBTS) are compared.

S_DLLBTS, CBLBTS, and FBLBTS have the similar implementation complexity because all of them have only single level sorting unit which dominates the implementation complexity, while *I_DLLBTS* has a double implementation complexity due to its two levels sorting units.

The objective of this study is to demonstrate that both *I_DLLBTS* and *S_DLLBTS* can protect the conforming traffic from the impact of the malicious flows while guarantee the zero class-based violation. The performance metrics used throughout this evaluation are the average packet delays of flows and the class-based violation at the output port.

Before proceeding with the presentation of the results of our comparative study, the experimental scenarios and the specific details of the traffic characteristics of the individual flows and classes considered in this evaluation are described. Specifically, there are 8 groups of bursty traffic, that belong to 4 different classes. Each group consists of 100 flows. The flow traffic shaping parameters are same for each flow: Peak Rate=16000 bytes/sec, Sustainable Rate=12000 bytes/sec, and their associated burst sizes, Peak Burst Size=15000 bytes, and Committed Burst Size=80000 bytes respectively. The length of packets varies from 40 bytes to 1500 bytes. Group 0 and group 1 belong to class 0, group 2 and group 3 belong to class 1, group 4 and group 5 belong to class 2, and group 6 and group 7 belong to class 3. The outlink rate is OC3, 155.52Mbps, or 19.44Mbytes/sec. The length of the byte slot is the reciprocal of the outlink byte rate.

The class traffic shaping parameters are also same for all classes: Peak Rate=3300000 bytes/sec, Sustainable Rate=2500000 bytes/sec, Peak Burst Size=15000 bytes, Committed Burst Size=80000 bytes. The class traffic shaping parameters are used to control the aggregate behavior of a class. The behavior of each traffic shaper architecture was investigated under two scenarios: one is that all flows are conforming to the traffic contracts, and the other one is that all flows of group 0 are not conforming to the traffic contracts, while the remaining flows are still conforming. In this study, we observed the average delays for different groups and the class-based violation for each group. Because only the aggregate behavior is monitored in the DiffServ networks, only low class-based violation should be guaranteed. The class-based violation may lead the violated traffic dropped in the DiffServ networks.

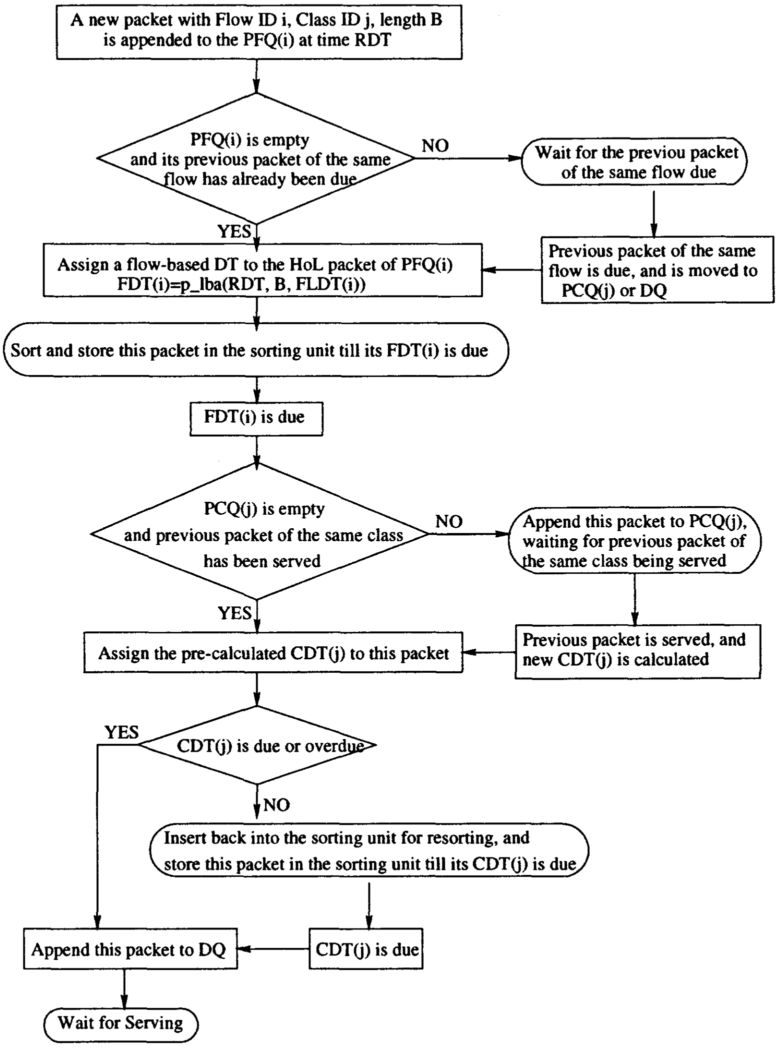
Fig. 6.7 presents the average packet delay of each group under the scenario that all flows are conforming, for all the four different architectures.

Fig. 6.7 presents the average packet delay of each group under the scenario that all flows are conforming, for all the four different architectures. From this figure it is observed that in each scheme, all groups have similar delay performance when all flows are conforming to the corresponding traffic contracts. The FBLBTS has lower delays, but it leads to high class-based violation ratio, 4.80% as can be seen in Fig. 6.9, while the other schemes give zero violation ratio.

Similarly, Fig. 6.8 presents the corresponding results under the scenario that the flows of the group 0 are non-conforming to their contracts.

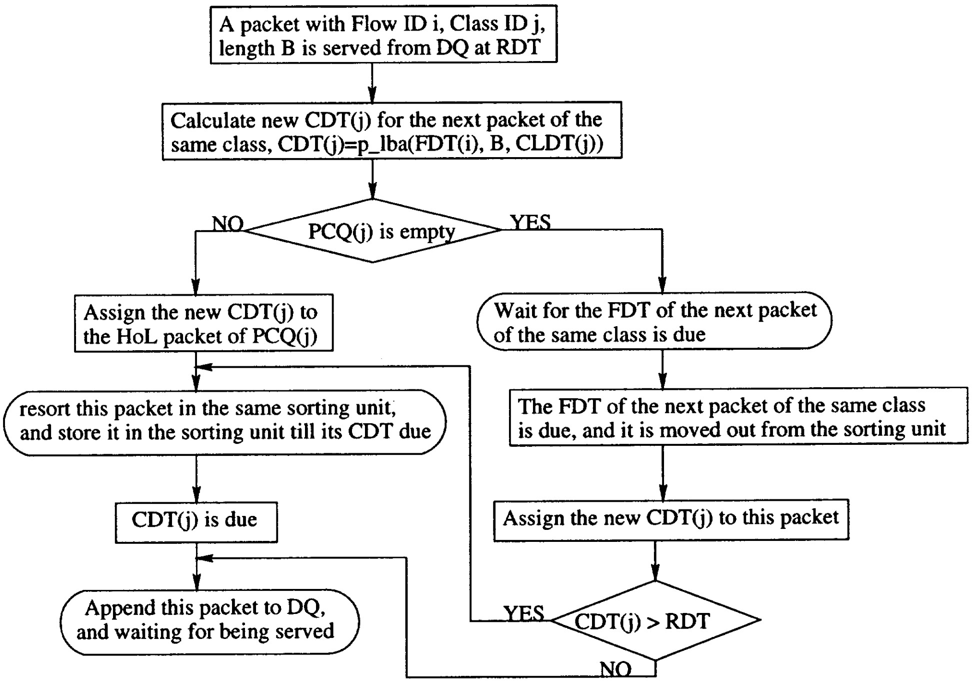
Fig. 6.8(a) shows that although only flows of the group 0 are non-conforming, the performance of both group 0 and group 1 is degraded significantly. The reason is that the CBLBTS can not differentiate the flows belonging to the same class. Therefore once some flows are non-conforming, all flows in the same class will be affected, and therefore those misbehaving flows will impact negatively the QoS of other conforming flows of the same class.

Fig. 6.8(b)(c)(d) show that FBLBTS, I_DLLBTS, S_DLLBTS can all isolate the non-conforming flows from affecting the performance of conforming traffic belonging to the same class. In each of these three schemes, all conforming traffic has almost same performance in both cases, and only the non-conforming flows are punished heavily. The S_DLLBTS scheme has a similar performance to the FBLBTS. However, FBLBTS can not guarantee the aggregate behavior of a class. Its class-based violation is very high, as shown in Fig. 6.9, which is 4.82%, and therefore it is not suitable for the DiffServ networks. Comparing the Fig. 6.8(c) and (d), it is found that although the both schemes have similar performance for those conforming traffic, the *I_DLLBTS* does have a better performance for the non-conforming traffic because in *I_DLLBTS*, the non-conforming traffic can occupy available bandwidth in the class after the conforming one occupied its share. While the I_DLLBTS has high implementation complexity, and may not be a practical implementation architecture. S_DLLBTS reduces the implementation complexity only at the expense of the degradation of the performance of the non-conforming traffic. It can be concluded that the S_DLLBTS can realize both flow-based and class-based traffic control while maintains low implementation complexity.


A new packet with Flow ID i, Class ID j, length B is appended to the PFQ(i) at time RDT
PFQ(i) is empty and its previous packet of the same flow has already been due
NO
Wait for the previou packet of the same flow due
YES
Assign a flow-based DT to the HoL packet of PFQ(i) FDT(i)=p_lba(RDT, B, FLDT(i))
Previous packet of the same flow is due, and is moved to PCQ(j) or DQ
Sort and store this packet in the sorting unit till its FDT(i) is due
FDT(i) is due
PCQ(j) is empty and previous packet of the same class has been served
NO
Append this packet to PCQ(j), waiting for previous packet of the same class being served
YES
Assign the pre-calculated CDT(j) to this packet
Previous packet is served, and new CDT(j) is calculated
YES
CDT(j) is due or overdue
NO
Insert back into the sorting unit for resorting, and store this packet in the sorting unit till its CDT(j) is due
Append this packet to DQ
CDT(j) is due
Wait for Serving

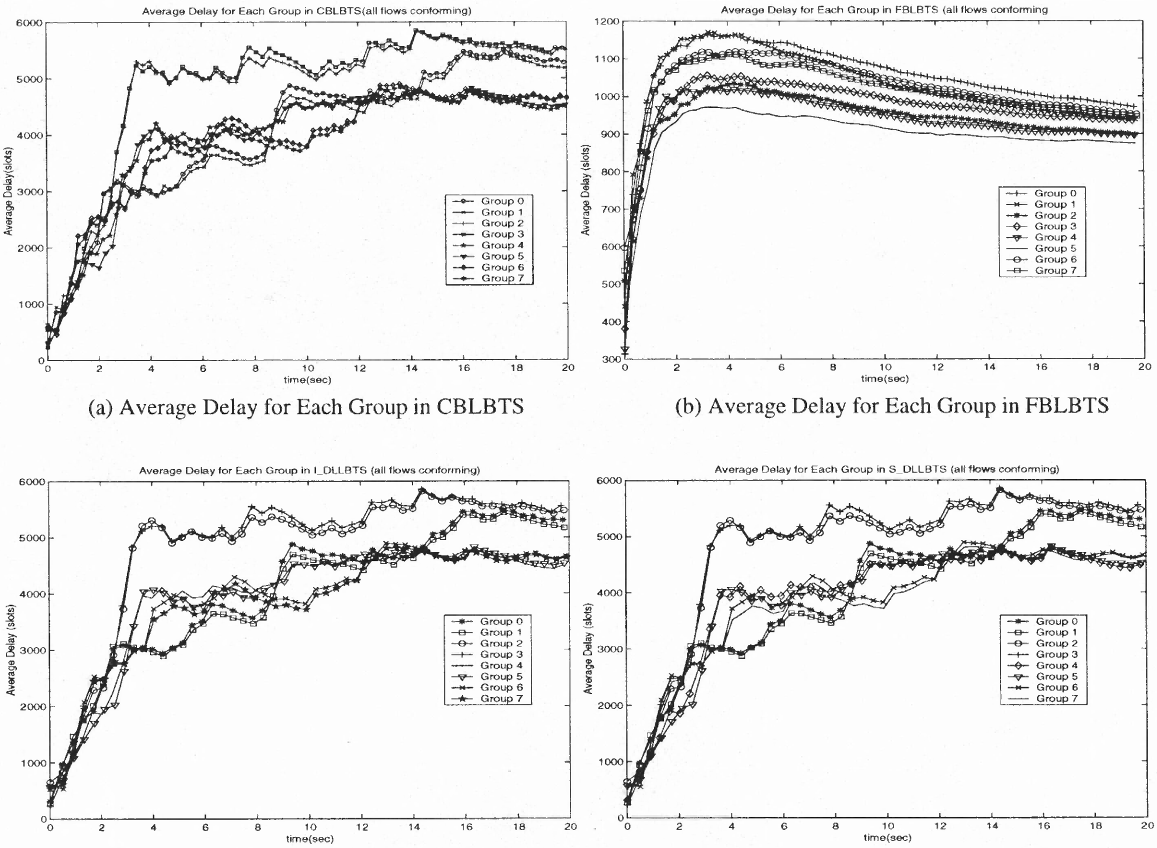
FLDT(i), FDT(i): Last Departure Time and Departure Time of Flow i;
CLDT(j), CDT(j): Last Departure Time and Departure Time of Class j;
p_lba: Packet version Leaky Bucket Algorithm;
PCQ(j): Per-Class-Queue of class j; PFQ(i): Per-Flow-Queue of flow i

Figure 6.5 The Operation Process Triggered by The Arrival of A Packet


A packet with Flow ID i, Class ID j, length B is served from DQ at RDT
Calculate new CDT(j) for the next packet of the same class, CDT(j)=p_lba(FDT(i), B, CLDT(j))
NO
PCQ(j) is empty
YES
Assign the new CDT(j) to the HoL packet of PCQ(j)
Wait for the FDT of the next packet of the same class is due
resort this packet in the same sorting unit, and store it in the sorting unit till its CDT due
The FDT of the next packet of the same class is due, and it is moved out from the sorting unit
CDT(j) is due
Assign the new CDT(j) to this packet
Append this packet to DQ, and waiting for being served
YES
CDT(j) > RDT
NO

FLDT(i), FDT(i): Last Departure Time and Departure Time of Flow i;
CLDT(j), CDT(j): Last Departure Time and Departure Time of Class j;
p_lba: Packet version Leaky Bucket Algorithm;
PCQ(j): Per-Class_Queue of class j; PFQ(i): Per-Flow_Queue of flow i

Figure 6.6 The Operation Process Triggered by The Departure of A Packet



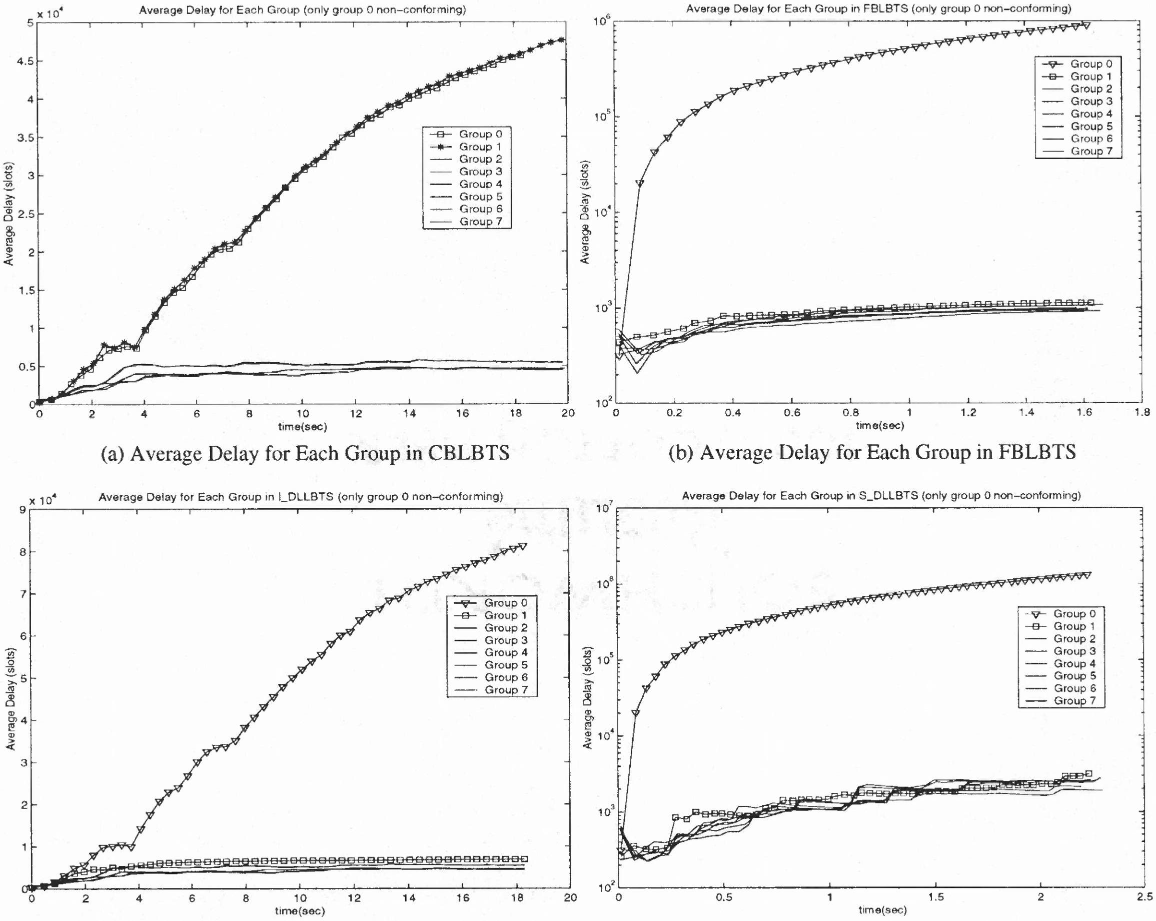
(a) Average Delay for Each Group in CBLBTS

(b) Average Delay for Each Group in FBLBTS

(c) Average Delay for Each Group in I_DLLBTS

(d) Average Delay for Each Group in S_DLLBTS

Figure 6.7 The Average Packet Delay of Each Group (all flows conforming)



(a) Average Delay for Each Group in CBLBTS

(b) Average Delay for Each Group in FBLBTS

(c) Average Delay for Each Group in I_DLLBTS

(d) Average Delay for Each Group in S_DLLBTS

Figure 6.8 The Average Packet Delay of Each Group (flows of group 0 non-conforming)

		All Flows Conforming	Group 0 Non-conforming
Class-based Violation Ratio	CBLBTS	0	0
	FBLBTS	4.80%	4.82%
	I_DLLBTS	0	0
	S_DLLBTS	0	0

CBLBTS: single level Class-Based Leaky Bucket Traffic Shaper; FBLBTS: single level Flow-Based Leaky Bucket Traffic Shaper;

I_DLLBTS: Ideal Dual Level Leaky Bucket Traffic Shaper; S_DLLBTS: Simplified Dual Level Leaky Bucket Traffic Shaper.

(Violation Ratio = Number of Violation Packets/Number of the Total Number of Output Packets)

Figure 6.9 Class-Based Violation Comparison in All Schemes

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

In order to utilize the network resources effectively while providing a wide range of satisfactory QoS guarantees, QoS mechanisms should be deployed. This dissertation studies the traffic shaping mechanism, which is one of the most important QoS mechanisms, and proposed algorithms and implementation architectures for it.

A novel hierarchical sorting unit is proposed to simplify the implementation of the time stamp sorting in the traffic shaper of a high speed packet switch. By utilizing different scheduling algorithms, it can be implemented as a Hierarchical Exact Sorting Traffic Shaper, or a Hierarchical Rate-Adaptive Scheduler. When it is used as a traffic shaper, it can implement exact sorting whose implementation reduces the number of timing queues substantially. In comparison with the rate-based grouping traffic shaper, it avoids the implementation of the complex scheduler, while keeping the comparable number of timing queues. Although the rate-based grouping traffic shaper reduces the number of sorting queues, its performance for low rate connection degrades greatly. Additionally, it requires hardware reconfigurations since the number of groups depends on the continuously changing traffic distribution. The traffic shaper with hierarchical timing queues maintains the good performance for all connections as the traffic shaper with direct exact sorting architecture but reduces the implementation complexity significantly.

When the architecture is used as a hierarchical rate-adaptive scheduler, it reduces the Cell Transfer Delay and maximum memory occupancy while taking other QoS requirements and fairness into consideration. The most distinctive feature is that it can work as a shaper or a scheduler based on the traffic load: When the traffic load is heavy, it functions as a shaper, and when the traffic load is light, it performs the GCRA scheduling function. In addition, by introducing a FIFO queue to accommodate the best effort traffic in the hierarchical sorting unit, we can implement a near work-conserving scheduler without increasing the loss ratio for the QoS traffic. In this way, the proposed scheduler can not only improves the link utilization, but also guarantees QoS requirements for the RSVP traffic. This novel hierarchical sorting unit's simplicity of implementation, good performance for very diverse traffic parameters, flexible feature, and independence of implementation structure on the traffic distribution make it practical and attractive for high speed packet switch designers.

An improved fair traffic shaping algorithm, called DED+CTS is proposed. It modifies the DEDTS algorithm by adding a process to resort the CT of those conforming cells. This algorithm can naturally give those high speed connections higher priority while guarantee their conformance. Consequently, the serious performance degradation of the high speed connections due to the loss of contentions in the heavy traffic load is avoided without increasing total available bandwidth. The performance improvement of those high speed connections only introduces slight performance degradation of those low speed connections. This algorithm can be implemented easily by replacing the FIFO DQ of the HTQTS with a small sorting unit which can be a sequencer.

The DiffServ framework is proposed to support QoS requirements in the Internet. In order to provide the QoS requirements, the access control at the edge of the DiffServ network is necessary. DLLBTS architecture is proposed to meet this requirement. This architecture can guarantee the traffic conforming to both flow-based and class-based traffic regulation when flows merge into different classes at the DiffServ network boundary. A practical implementation architecture is also proposed for the DLLBTS scheme, named *S_DLLBTS*. It can be implemented efficiently with off the shelf hardware technology. By the simulation results, we can see that this architecture can successfully realize not only the flow-based traffic control, but also the class-based traffic shaping. It is a practical traffic shaper architecture for the edge nodes of DiffServ networks.

The following issues should be further studied. The studies of this dissertation concentrate on the traffic shaping under lossless scenario. In the future work, the buffer management should be taken into consideration for the loss of traffic due to the shortage of buffer space, which exists in the real system. Strategies for congestion detection and traffic dropping should be combined into the design of the traffic shaping algorithms and implementations. For the proposed improved fair traffic shaping algorithm, the further theoretical analysis will be performed. In addition, the proposed dual level access control architecture only considers the bandwidth allocation, no priorities for different classes are considered. In the further study, scheduling algorithms should be implemented at the output of the dual level access control architecture to differentiate the QoS requirements for different classes which is one goal of the DiffServ networks.

APPENDIX A

NUMBER OF THE BACKLOGGED CONFORMING CELLS IN A SHAPING MULTIPLEXER

In the previous discussion, it concludes that there is a correlation between the rate of a connection and the length of DQ. When the number of the backlogged conforming cells is larger than the incremental interval I of the leaky bucket of a connection, then the leaky bucket loses its function, in which I is the reciprocal of the committed rate of the connection. In this appendix, it shows that the number of the backlogged conforming cells in a shaping multiplexer can be larger than the incremental interval of a leaky bucket associated with a connection as the admitted traffic increasing.

Elwalid and Mitra have analyzed the lossless multiplexing in a shaping multiplexer in [30], and gave a tight bound for the number of backlogged cells and the number of admissible connections. Here, the analysis in [30] is briefly reviewed, and a lower bound of the number of admissible connections which will lead to the loss of the function of a leaky bucket is derived.

The single source shaping system and the multiple source shaping multiplexer are shown in Fig. A.1 and Fig. A.2, respectively.

In Fig. A.1, the shaping parameters for one connection is (r, B_t, P), in which, P is the PCR, r is the SCR, and B_t is the limit of the leaky bucket representing the maximum delay variance. The actual bandwidth occupied by this connection is denoted by c $(r < c < P)$, and the number of present available token is denoted by x(t). Let $A(t, t+\tau)$ denote the total cells in any interval $[t, t+\tau]$, then,

$$A(t, t+\tau) \leq min(P\tau, x(t) + r\tau) \ \forall t, \ \forall \tau \qquad (A.1)$$

A proposition for the buffer size of single source shaping system y(t) is given in [30],

$$y(t) \leq \frac{(P-c)B_t}{(P-r)}, \ for \ all \ t \qquad (A.2)$$

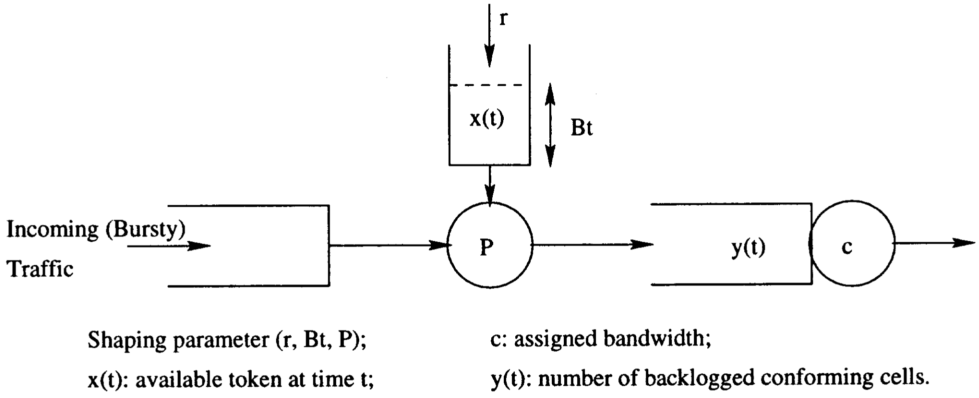

r
x(t)
Bt
Incoming (Bursty)
Traffic
P
y(t)
c
Shaping parameter (r, Bt, P);
x(t): available token at time t;
c: assigned bandwidth;
y(t): number of backlogged conforming cells.

Figure A.1 Single Source Shaping System Model

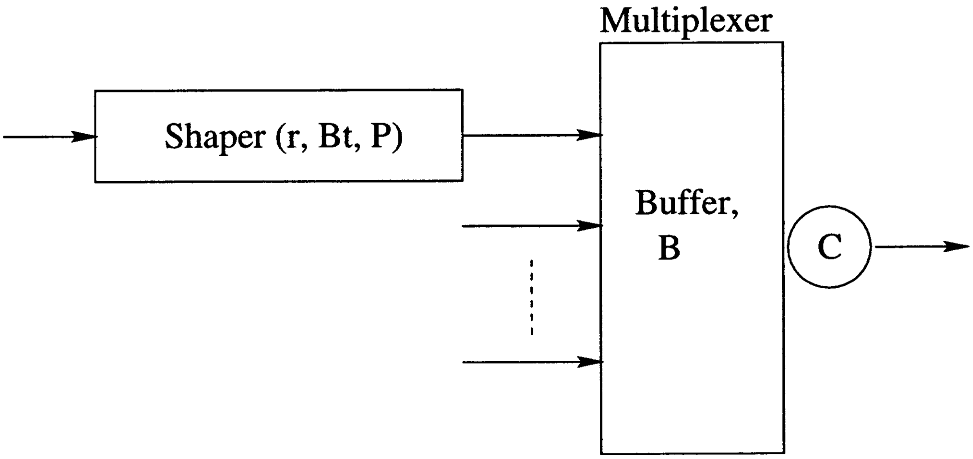


Figure A.2 Multiple Source Shaping Multiplexer

Fig. A.2 depicts the shaping multiplexer with multiple sources. The total buffer size of the multiplexer is denoted by B, and the total output link capacity is denoted by C. Elwalid and Mitra considered a multiplexer with K shaped traffic streams, which all have same shaping parameter (r, B_t, P) [30]. Based on the single source shaping system result, they derive a tight bound on the content of the multiplexing buffer in the shaping multiplexer, Y(t),

$$Y(t) \leq \frac{(KP - C)B_t}{P - r}, \; for \; all \; t \tag{A.3}$$

Hence, for lossless multiplexing, the buffer size B should satisfy the inequality,

$$B \geq \frac{(KP - C)B_t}{P - r} \tag{A.4}$$

From this bound, the lower bound for number of the accepted connections, K, can be derived so that the buffer size is larger than the 1/r, which is the incremental interval of the leaky buckets associated with these connections.

$$\frac{(KP - C)B_t}{P - r} \geq \frac{1}{r} \tag{A.5}$$

From this inequality, we get when

$$K \geq \frac{1}{rB_t} - \frac{1}{PB_t} + \frac{C}{P}, \tag{A.6}$$

the buffer size can reach the 1/r, and as the K increases, the buffer size will increase linearly. Higher K means more accepted connections and heavier load, therefore we can conclude that the length of the output queue can be larger than 1/r. In the scenario of a multiplexer with heterogeneous sources, similar results can be derived. The shaping parameters for connection i are denoted by (r_i, B_{ti}, P_i), then we can get the inequality

$$Y(t) = \sum_{i=1} K y_i(t) \qquad \leq \textstyle\sum_{i=1} K \frac{(P_i - c_i)B_{ti}}{P_i - r_i} \tag{A.7}$$

$$\leq K \max_{i \in K}\left(\tfrac{(P_i - c_i)B_{ti}}{P_i - r_i}\right). \tag{A.8}$$

These inequalities give the bounds of the length of the DQ. They can give us the clue in design the size of the sequencer.

APPENDIX B

CONFORMING DEPARTURE TIME BASED PGPS

Parekh and Gallager proposed Generalized Processor Sharing (GPS) approach in [69] to guarantee the fair sharing of the output link between competing connections. It's the basis of the packet scheduler. It divides output link among K competing connections proportional to a set of weights, $(\phi_0, \phi_1, ..., \phi_i, ..., \phi_K)$. $(r_0, r_1, ..., r_i, ..., r_K)$ denotes the rate requirements of the K connections, then, when $\phi_i \propto r_i$, GPS can guarantee

1. The bandwidth occupancy of connection i to be r_i, independent of other connections' requirements;

2. The delay bound of the connection i only related to its own queue length, independent of other connections' queue [69].

However, GPS can not be implemented because it requires the server can serve all connections at the same time as if all traffic can interleave with each other as a fluid model. The *packet-by-packet* GPS (PGPS) was proposed to simulate GPS [69], and it proved that for all times τ and connection i:

$$S_i(0, \tau) - \hat{S}_i(0, \tau) \leq L_{max}, \qquad (B.1)$$

in which, $S_i(\tau, t)$ and $\hat{S}_i(\tau, t)$ are the amount of connection i traffic served under GPS and PGPS [69]. Parekh and Gallager proposed *Virtual Time Implementation* of PGPS [69]. In this virtual time system, a virtual time $V(t)$ is maintained as

$$V(0) = 0 \qquad \text{(B.2)}$$

$$\frac{dV(t)}{dt} = \frac{1}{\sum_{i \in B_j} \phi_i}, \qquad \text{(B.3)}$$

in which, B_j is the number of active connections. Suppose that the Kth connection i packet with length L_i^k arrives at time a_i^k, then the virtual times at which the packet begins and finishes service (denoted by S_i^k and F_i^k respectively) are defined as below, and following the notations in [69]

$$S_i^k = max(F_i^{k-1}, V(\alpha_i^k)) \tag{B.4}$$

$$F_i^k = S_i^k + \frac{L_i^k}{\phi_i} \tag{B.5}$$

When a packet enters the PGPS system, it is assigned a starting service time. If it is the HoL packet of this connection, then its starting service time is the current virtual system time $V(\alpha_i^k)$, otherwise, the starting service time is the finishing service time of the last packet in the same connection. At the same time, the finishing time for this packet is assigned as the sum of the starting service time and the service time. The server will pick the packet with the smallest finishing time to serve when multiple packets compete for the service.

The algorithm DED+CTS proposed in the dissertation is a special case of the virtual time PGPS implementation. It choose the conforming departure times (DTs) of cells as the virtual time, and sustainable cell rates as the weights in the bandwidth sharing among competing connections, and all packets are cells with fixed length 1, so this conforming departure time based PGPS can be represented as

$$S_i^k = DT_i^k \tag{B.6}$$

$$F_i^k = S_i^k + \frac{1}{\rho_i} \tag{B.7}$$

Therefore, DED+CTS inherits all characteristics of the PGPS, implementing the fair scheduling among the backlogged cells, while keeping all cells strictly conforming to the traffic contracts. Consequently, we can conclude that it is a fair traffic shaping algorithm.

REFERENCES

1. S. Ano, T. Hasegawa, and T. Kato. "A Study on Accommodation of TCP/IP Best-Effort Traffic to Wide Area ATM Network with VBR Service Category Using Selective Cell Discard". *IEEE Proceedings of ATM Workshop'99*, May 1999.

2. J. Bennett and H. Zhang. "Hierarchical Packet Fair Queueing Algorithms". *IEEE/ACM Transactions on Networking*, 5, October 1997.

3. J. Beran, R. Sherman, M. Taqqu, and W. Willinger. "Long-Range Dependence in Variable-Bit-Rate Video Traffic". *IEEE Transactions on Communications*, 43, Feb.-April 1995.

4. Y. Bernet. "The Complementary Roles of RSVP and Differentialted Services in the Full-Service QoS Network". *IEEE Communications Magazine*, February 2000.

5. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. "An Architecture for Differentiated Services". *IETF RFC 2475*, December 1998.

6. O. Bonaventure and S. D. Cnodder. "A Rate Adaptive Shaper for Differentiated Services". *IETF RFC 2963*, October 2000.

7. P. Boyer, F. M. Guillemin, M. J. Servel, and J. P. Coudreuse. "Spacing Cells Protects and Enhances Utilization of ATM Network Links". *IEEE Network*, 6, May 1992.

8. P. E. Boyer, M. J. Servel, and F. P. Guillemin. "The Spacer-controller: An Efficient UPC/NPC for ATM Networks". *International Switching Symposium*, 2, October 1992.

9. R. Braden, D. Clark, and S. Shenker. "Integrated Services in the Internet Architecture: an Overview". *Internet RFC 1633*, June 1994.

10. U. Briem, E. Wallmeier, C. Beck, and F. Matthiesen. "Traffic Management for an ATM Switch with Per-VC Queueing:Concept and Implementation". *IEEE Communications Magazine*, Jan. 1998.

11. M. Casoni and J. S. Turner. "On the Performance of Early Packet Discard". *IEEE Journal on Select Areas in Communications*, 15, June 1997.

12. H. J. Chao. "Design of Leaky Bucket Access Control Schemes in ATM Networks". *Proceedings of ICC'91*, June 1991.

13. H. J. Chao. "Design of Leaky Bucket Access Control Schemes in ATM Networks". *Conference Record of the ICC, Denver, CO*, June 1991.

14. H. J. Chao, H. Cheng, Y. R. Jenq, and D. Jeong. "Design of a Generalized Priority Queue Manager for ATM Switches". *IEEE Journals on Selected Areas on Communications*, 15, June 1997.

15. H. J. Chao and J. S. Hong. "Design of an ATM shaping multiplexer with guaranteed output burstiness". *Intl. Journal of Computer System Science & Engineering, Special issue on ATM Switching*, 12(2):131–141, Mar. 1997.

16. H. J. Chao and I. H. Pekcan. "Priority Cell Departing and Discarding in ATM Switch Nodes". *Proceedings of IEEE Workshop in High-Performance Communications Subsystems*, Sept. 1993.

17. H. J. Chao and D. E. Smith. "Design of Virtual Channel Queue in an ATM Broadband Terminal Adaptor". *Proceedings of INFOCOM'92*, May 1992.

18. H. J. Chao and N. Uzun. "A VLSI Sequencer Chip of ATM Traffic Shaper and Queue Manager". *IEEE Journal of Solid-State Circuits*, 27(11), Nov. 1992.

19. H. J. Chao and N. Uzun. "An ATM Queue Manager with Multiple Delay and Loss Priorities". *IEEE/ACM Transactions on Networking*, 3, December 1995.

20. W. Chen, R. Lee, and H. Lin. "A Novel Cell Scheduler with QoS Guarantee for Services in ATM Networks". *IEICE TRANSACTION COMMUNICATION*, E82-B(2), February 1999.

21. W. Chen and H. T. Mouftah. "Design and Analysis of an ABR Flow Control Protocol". *Proceedings of IEEE 1997 Canadian Conference on Electrical and Computer Engineering*, May 1997.

22. K. Cheon and S. S. Panwar. "On the Performance of ATM-UBR with Early Selective Packet Discard". *Proceedings of ICC'98*, June 1998.

23. F. M. Chiussi and A. Francini. "Implementing Fair Queueing in ATM Switches: The Discrete-Rate Approach". *Proceedings of IEEE INFOCOM'97*, 1997.

24. A. K. Choudhury and E. L. Hahne. "Dynamic Queue Length Thresholds for Shared Memory Packet Switches". *IEEE/ACM Transactions On Networking*, 6, April 1998.

25. D. Clark. "The Design Philosophy of the DARPA Internet Protocol". *ACM SIGCOMM'88*, Aug. 1988.

26. S. D. Cnodder. "Rate Adaptive Shapers for Data Traffic in DiffServ Networks". *NetWorld+Interop 2000 Engineers Conference*, May 2000.

27. K. Dimyati and Y. T. Chin. “Policing Mechanism and Cell Loss Priority Control on Voice Cells in ATM Networks Using Fuzzy Logic”. *IEE Proceedings on Communications*, 147, Aug. 2000.

28. L. Dittmann, S. B. Jacoben, and K. Moth. “Flow Enforcement Algorithms for ATM Networks”. *IEEE Journal on Selected Areas in Communications*, April 1991.

29. A. Elwalid and D. Mitra. ”Analysis, Approximations And Admission Control of A Multi-Service Multiplexing System with Priorities”. *Proceeding of Infocom 95*, pages 463–472.

30. A. Elwalid and D. Mitra. “Traffic Shaping at a Network Node: Theory, Optimum Design, Admission Control”. *Proceedings of INFOCOM'97*, 2, March 1997.

31. A. I. Elwalid and D. Mitra. “Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks”. *IEEE/ACM Transactions on Networking*, 1(3), Jun. 1993.

32. D. Awduche et al. “Extension to RSVP for Traffic Engineering”. *Internet draft, draft-swallow-mpls-RSVP-trafeng-00.txt*, Aug. 1998.

33. C. Fang and A. Lin. “On TCP Performance of UBR with EPD and UBR-EPD with a Fair Buffer Allocation Scheme”. *ATM Forum Contribution 95-1645*, December 1995.

34. The ATM Forum. Traffic management specification version 4.0. Apr. 1996.

35. P. Goyal, H. M. Vin, and H. Cheng. “Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks”. *IEEE/ACM Transactions on Networking*, 5, October 1997.

36. R. Goyal, R. Jain, S. Kalyanaraman, S. Fahmy, and S. Kim. “UBR+ Improving Performance of TCP over ATM-UBR Service”. *Proceedings on ICC'97*, June 1997.

37. R. Guérin, H. Ahmadi, and M. Naghshineh. “Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks”. *IEEE J. Select. Areas Commun.*, 9(7):968–981, Sep. 1991.

38. R. Guerin, S. Blake, and S. Herzog. “Aggregating RSVP-based QoS Requests”. *Internet draft, draft-guerin-aggreg-RSVP-00.txt*, Nov. 1997.

39. F. Guillemin, P. Boyer, A. Dupuis, and L. Romoeuf. “Peak Rate Enforcement in ATM Networks”. *IEEE INFOCOM'92*, 2, May 1992.

40. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. “Assured Forwarding PHB Group”. *IETF RFC 2597*, June 1999.

41. J. Heinanen and R. Guerin. "A Single Rate Three Color Marker". *IETF RFC 2697*, September 1999.

42. J. Heinanen and R. Guerin. "A Two Rate Three Color Marker". *IETF RFC 2698*, September 1999.

43. M. G. Hluchyj and N. Yin. "A Second-Order Leaky Bucket Algorithm to Guarantee QoS in ATM Networks". *Proceedings of IEEE Globecom'96*, December 1996.

44. J. S. Hong. *"Design of an ATM shaping multiplexer algorithm and architecture"*. Ph.D. Thesis. Polytechnic University, Aug. 1996.

45. C. Hsu, A. Ortega, and A. R. Reibman. "Joint Selection of Source and Channel Rate for VBR Video Transmission Under ATM Policing Constraints". *IEEE Journal on Selected Areas in Communications*, 15, August 1997.

46. ITU-T Rec. I.371. "Traffic Control and Congestion Control in B-ISDN". March 1995.

47. K. Iida, T. Takine, H. Sunahara, and Y. Qie. "Delay Analysis for CBR Traffic in Static-Priority Scheduling: Single-Node and Heterogeneous CBR Traffic Case". *Proceedings of GLOBECOM'98*, Nov. 1998.

48. F. Ishizaki, T. Takine, and Y. Oie. "Delay Analysis for Real-Time and Non Real-Time Traffic Streams under a Priority Cell Scheduling". *Proceedings of GLOBECOM'98*, Nov. 1998.

49. L. Jaussi, M. Lorang, and J. Nelissen. "A Detailed Experimental Performance Evaluation on TCP over UBR". *Proceedings of 1st IEEE International Conference on ATM*, June 1998.

50. T. Kalkowski, W. Burakowski, and A. Manikowski. "Effectiveness of the ATM Services for the TCP Protocol Applications". *Proceedings of MILCOM'96*, Oct. 1996.

51. K. Kawahara, K. Kitajima, T. Takine, and Y. Oie. "Packet Loss Performance of Selective Cell Discard Scheme in ATM Switches". *IEEE Journal on Select Areas in Communications*, 15, June 1997.

52. R. Kawamura, H. Hadama, K. Sato, and I. Tokizawa. "Fast VP-Bandwidth Management with Distributed Control in ATM Networks". *IEICE Trans. on Communications*, E77-B, January 1994.

53. G. Kesidis, J. Walrand, and C. Chang. "Effective Bandwidths for Multiclass Markove Fluids and Other ATM Sources". *IEEE Trans. on Networking*, 1(4):424–428, Aug. 1993.

54. D. Kim, C. Park, and Y. Cho. “Impacts of Fast Congestion Notification Function on Multicast ABR Branch Point Switch in ATM Networks”. *Proceedings of ICC'2000*, 3, June 2000.

55. D. Kim, Y. Park, and J. Kim. “Point to Multipoint ABR Flow Control in ATM Networks”. *Proceedings of the IEEE Conference on High Performance Switching and Routing'2000*, June 2000.

56. T. Kim, B. Roh, and J. Kim. “Bandwidth Renegotiation with Traffic Smoothing and Joint Rate Control for VBR MPEG Video over ATM”. *IEEE Transactions on Circuits and Systems for Video Technology*, 10, Aug. 2000.

57. E. W. Knightly. “H-BIND: A New Approach to Providing Statistical Performance Guarantees to VBR Traffic”. *Proceedings of IEEE INFOCOM'96*, March 1996.

58. E. W. Knightly and N. B. Shroff. “Admission Control for Statistical QoS: Theory and Practice”. *IEEE Network*, 13, March 1999.

59. E. W. Knightly and H. Zhang. “D-BIND: An Accurate Traffic Model for Providing QoS Guarantees to VBR Traffic”. *IEEE/ACM Transactions on Networking*, 5, April 1997.

60. B. Lague, C. Rosenberg, and F. Guillemin. “A Generalization of Some Policing Mechanisms”. *Proceedings of IEEE INFOCOM'92*, 2, May 1992.

61. T. V. Lakshman, A. Ortega, and A. R. Reibman. “VBR Video: Tradeoffs and Potentials”. *Proceedings of The IEEE*, 86, May 1998.

62. Y. Levy and X. Mang. “A New QoS-Driven Packet Discarding Scheme”. *Proceedings of IEEE GLOBECOM'99*, 2, December 1999.

63. T. Li and Y. Rekhter. “Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)”. *RFC 2430*, Oct. 1998.

64. T. Lizambri, F. Duran, and S. Wakid. “Priority Scheduling and Buffer Management for ATM Traffic Shaping”. *Proceedings of 7th IEEE Workshop on Future Trends of Distributed Computing Systems*, Dec. 1999.

65. A. Mehaoua, R. Boutaba, S. Pu, Y. Rasheed, and A. Leon-Garcia. “Towards an Efficient ATM Best Effort Video Delivery Service”. *Proceedings of ICC'99*, May 1999.

66. K. Nichols, S. Blake, F. Baker, and D. Black. “‘Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”. *IETF RFC 2474*, Dec. 1998.

67. K. Nichols, V. Jacobson, and L. Zhang. "A Two-Bit Differentiated Services Architecture for the Internet". *IETF RFC 2638*, July 1999.

68. D. Pao. "A Congestion Control Algorithm for Multipoint-to-Multipoint ABR Service in ATM Network". *Proceedings of IEEE Conference on High Performance Switching and Routing'2000*, June 2000.

69. A. K. Parekh and R. G. Gallager. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case". *IEEE/ACM Transactions on Networking*, 1, June 1993.

70. A. K. Parekh and R. G. Gallager. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case". *IEEE/ACM Transactions on Networking*, 2, April 1994.

71. E. P. Rathgeb. "Modeling and Performance Comparison of Policing Mechanisms for ATM Networks". *IEEE Journal on Selected Areas in Communications*, 9, April 1991.

72. E. P. Rathgeb. "Modeling and Performance Comparison of Policing Mechanisms for ATM Networks". *IEEE Journal on Selected Areas in Communications*, SAC-9(1), April 1991.

73. J. Rexford, F. Bonomi, A. Greenberg, and A. Wong. "Scalable architectures for integrated traffic shaping and link scheduling in high-speed ATM switches". *IEEE Journal on Selected Areas in Communications*, pages 938–950, Jun. 1997.

74. D. Saha, S. Mukherjee, and S. K. Tripathi. "Carry-Over Round Robin: A Simple Cell Scheduling Mechanism for ATM Networks". *IEEE/ACM Transactions on Networking*, 6, December 1998.

75. H. Saito and K. Shiomoto. "Dynamic Call Admission Control in ATM Networks". *IEEE J. Select. Areas Commun.*, 9(7):982–989, Sep. 1991.

76. S. Shenker, C. Partridge, and R. Guerin. "Specification of Guaranteed Quality of Service". *RFC 2212*, Sept. 1997.

77. T. Sheu and K. Lee. "A Multiple-Class Buffer Allocation Scheme for ATM Networks with VBR Traffic". *Proceedings of ICOIN'98*, 1998.

78. K. Shim, J. Nho, and J. Lim. "On priority Scheduling Algorithm at ATM Switches with Multi-Class Output Buffers". *IEICE TRANSACTION COMMUNICATION*, E82-B(1), January 1999.

79. M. Sidi, W. Z. Liu, I. Cidon, and I. Gopal. "Congestion Control through Input Rate Regulation". *IEEE Transactions on Communications*, 41, March 1993.

80. D. E. Smith and H. J. Chao. "Buffer Sizing at A Host in an ATM Network". *Proceedings of INFOCOM'92*, May 1992.

81. W. Stallings. *"High-Speed Networks–TCP/IP and ATM Design Principles"*. Prentice Hall, 1998.

82. D. Stiliadis and A. Varma. "A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms ". *Proceedings of IEEE INFOCOM'97*, 1997.

83. D. Stiliadis and A. Varma. "Efficient Fair Queueing Algorithms for Packet-Switched Networks". *IEEE/ACM Transactions on Networking*, April 1998.

84. D. Stiliadis and A. Varma. "Latency-Rate Servers: A General Models for Analysis of Traffic Scheduling Algorithms". *IEEE/ACM Transactions on Networking*, 6, October 1998.

85. I. Stoica, H. Zhang, and T. Ng. "A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time, and Priority Services". *IEEE/ACM Transactions on Networking*, 8, April 2000.

86. I. Stoica, H. Zhang, and T. Ng. "A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time, and Priority Services". *IEEE/ACM Transactions on Networking*, *, April 2000.

87. Y. T. Wang, T. P. Lin, and K. C. Gan. "An improved scheduling algorithm for weighted round-robin cell multiplexing in an ATM switch". *Proc. IEEE ICC'94*, 1994.

88. D. Wrege, E. Knightly, H. Zhang, and J. Liebeherr. "Deterministic delay bounds for VBR video in packet switching networks: Fundamental limits and practical trade-offs". *IEEE/ACM Transactions on Networking*, 4, June 1996.

89. J. Wroclawski. "Specification of the Controlled-Load Network Element Service". *RFC 2211*, Sept. 1997.

90. T. Wu and N. Yoshikai. *"ATM Transport and Network Integrity"*. Academic Press, 1997.

91. X. Xiao and L. M. Ni. "Internet QoS: A Big Picture". *IEEE Network*, March/April 1999.

92. N. Yamanaka, Y. Sato, and K. Sato. "Usage Parameter Control and Bandwidth Allocation Methods Considering Cell Delay Variation in ATM Networks". *IEICE Transaction on Communications*, March 1993.

93. N. Yin. "Rate-based Flow Control and It's Application in IP/ATM Internetworks". *Professional Program Proceedings on Electronics Industries Forum of New England, 1997*, May 1997.

94. S. Zeng and N. Uzun. "A Novel Hierarchical Traffic Shaper/Scheduler for High Speed Packet Switches". *submitted to IEEE transaction on Networking.*

95. S. Zeng and N. Uzun. "A Hierarchical Traffic Shaper for Packet Switches". *Proc. IEEE GLOBECOM'99*, December 1999.

96. S. Zeng, N. Uzun, and S. Papavassiliou. "A Dual Level Leaky Bucket Traffic Shaper Architecture for DiffServ Networks". *to be presented in 2001 IEEE Workshop on High Performance Switching and Routing*, May 2001.

97. S. Zeng, N. Uzun, and S. Papavassiliou. "An Improved Fair Traffic Shaping Algorithm for High Speed Packet Switches". *to be presented in 2001 IEEE Workshop on High Performance Switching and Routing*, May 2001.

98. S. Zeng, N. Uzun, and S. Papavassiliou. "Dual Level Access Control Architecture for QoS Management in Internet". *to be submitted to Journal of Network and Systems Management*, 2001.

99. S. Zeng, N. Uzun, M. Toy, and N. Gogate. "A Novel Hierarchical Rate-Adaptive Scheduler for High Speed Packet Switches". *NetWorld+Interop'2000, Las vegas, NV*, May 2000.

100. H. Zhang, O. W. Yang, and H. Mouftah. "The Hop-by-Hop Flow Controller for High-Speed Networks: Single VC Case". *Proceedings of GLOBECOM'97*, Nov. 1997.

101. L. Zhang, S.E. Deering, D. Estrin, and S. Shenker. "RSVP: A New Resorce ReSerVation Protocol". *IEEE Network*, 7(5):8–18, Sep. 1993.