Delay Guaranteed Cross Pushout and Loss Rate Differentiation for DiffServ Networks

submitted by

Ximing Zeng, B.Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute of Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering

Carleton University
Ottawa, Ontario, Canada K1S 5B6
September 20, 2004

@Copyright
2004, Ximing Zeng

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-97448-0

Our file Notre référence

ISBN: 0-612-97448-0

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

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

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

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

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

PAGINATION ERROR.

ERREUR DE PAGINATION.

TEXT COMPLETE.

LE TEXTE EST COMPLET.

Abstract

In DiffServ IP network, bandwidth and buffer space are allocated for flow aggregates to guarantee the Assured Forwarding (AF) service. If the input traffic characteristics are known a priori, the resource can be engineered precisely to meet the Quality of Service (QoS) requirements. However, in reality the input traffic is generally unpredictable due to bursty traffic in volume and due to the destinations of traffic flows being dynamic and random. As a result, a static bandwidth and buffer allocation can only guarantee average throughputs and delay bounds imposed by limited buffer sizes. The loss rate of each flow aggregate typically varies unless the network is over provisioned. In this thesis we develop a cross pushout scheme to allow one flow aggregate to “borrow” bandwidth and buffer space from other flow aggregates by selectively pushing out packets in other classes. We show that if all the packets have the same size in a system like ATM, with Delay Guaranteed Cross Pushout (DGCP) scheme, loss rates among flow aggregates can be adjusted according to the desired loss rate differentiation target without compromising the overall loss rate. We will also demonstrate that the mechanism guarantees that all packets can still meet their respective delay bounds. Furthermore, these delay bounds can still be guaranteed even if we extend DGCP to work under a system with variable packet size. As a result, with DGCP, a forwarding engine can not only provides delay bounds but also adjusts loss rates among service classes under the unpredictable dynamic traffic.

Acknowledgements

I would like to thank my thesis supervisor, Prof. Changcheng Huang and Prof. Anand Srinivasan for their guidance and encouragement throughout the course of my thesis. I appreciate their helpful suggestions and the patience they have shown to me during my study period here at the optical network research lab. I would also like to thank Dr. Mohammed Zaid and Dr. Robert Crawball of NCIT and Dr. Rama Munikoti and Dr. Kalai Chelvan of Eion Incorporation for their support when I worked as a research assistant in Eion Incorporation.

To my family

Table of Contents

Acceptance of the thesis.....	ii
Abstract.....	iii
Acknowledgements	iv
Table of Contents	vi
List of Tables	viii
List of Figures.....	ix
List of Notations	ix
List of Acronyms	xi
 1 Introduction	 1
1.1 Background.....	1
1.2 Thesis Contributions	5
1.3 Thesis Organization	6
 2 Differentiated Services	 7
2.1 Introduction.....	7
2.2 Core and Edge functions.....	8
2.2.1 Edge Functions.....	8
2.2.2 Core Functions	9
2.3 Per Hop Behaviors	9
2.3.1 Expedited Forwarding Class	9
2.3.2 Assured Forwarding Class	10
2.3.3 Best Effort Class	12
2.4 Scheduling Algorithms	12
2.4.1 Static Priority (SP) Scheduler	12
2.4.2 Earliest Deadline First (EDF) Scheduler	13
2.4.3 Fair Queueing Schedulers	15
2.5 QoS under Dynamic Traffic	22
 3 Proportional Loss Rate Differentiation Model.....	 25
3.1 Introduction.....	25
3.2 Queue Management Schemes on Providing Loss Rate Differentiation	26
3.2.1 Pushout.....	26

3.2.2 Partial Buffer Sharing (PBS)	27
3.2.3 Complete Buffer Partitioning (CBP)	27
3.2.4 Multi-class RED.....	28
3.3 Proportional Differentiated Service Model	29
4 Delay Guaranteed Cross Pushout	36
4.1 Design Principles of DGCP	37
4.2 Virtual Backlog.....	39
4.3 Delay Guarantee under DGCP.....	45
4.4 Total Loss under DGCP.....	52
4.5 Departure Sequence under DGCP	56
4.6 DGCP Implementation under WRR	60
4.6.1 Implementation of WRR Scheduler.....	63
4.6.2 Implementation of Per Class Logical Queues.....	65
4.6.3 Implementation of QM Schemes	66
4.6.4 Implementation of PLR Dropper	66
4.6.5 Implementation of DGCP	67
4.7 DGCP with Variable Packet Sizes.....	68
5 Simulation Result	80
5.1 Delay Guarantee under DGCP.....	81
5.2 Total Loss under DGCP.....	84
5.3 Delay Guarantee under DGCP+	87
5.4 Total Loss under DGCP+	90
6 Conclusions and Recommendations for Future Research....	91
6.1 Conclusions.....	91
6.2 Recommendations for Future Research.....	92
References	93

List of Tables

Table 2.1 Fairness under Fair Queueing Schedulers.	20
Table 2.2 Latency of Fair Queueing Schedulers.....	22
Table 5.1 QM Parameters in Experiment 1	81
Table 5.2 Simulation Parameters in Experiment 3.	88

List of Figures

Figure 3.1 The packet-forwarding engine of proportional differentiated service model..	30
Figure 3.2 A general packet-forwarding engine for AF service classes.	31
Figure 3.3 System 2: A DiffServ packet forwarding engine with PLR dropper.....	32
Figure 3.4 An example showing PLR dropper under fair queueing scheduler causes longer delay.....	34
Figure 3.5 An example showing PLR dropper causes more overall loss under fair queueing scheduler.....	35
Figure 3.6 An example showing the overall loss under a system without PLR dropper..	35
Figure 4.1 A example showing the bandwidth “borrowing” can guarantee the delay.	38
Figure 4.2 Procedure to update virtual backlog upone packet arrival.	41
Figure 4.3 Procedure to update the virtual backlog upon packet departure.....	42
Figure 4.4 An example showing how the virtual backlogs are updated.	45
Figure 4.5 System 3: A DiffServ packet forwarding engine with DGCP.....	46
Figure 4.6 A realization of real queue length $L_{1k}(t)$ or virtual backlog $L_{3k}(t)$	54
Figure 4.7 The architecture of a forwarding engine with DGCP under WRR scheduler.	63
Figure 4.8 Logical queue under DGCP.....	65
Figure 5.1 Simulated network topology.....	80
Figure 5.2 Loss rates under three systems.	82
Figure 5.3 CDFs of packets delay under three systems.	83
Figure 5.4 Tail part of CDFs of packets delay under three systems	84
Figure 5.5 Total loss under system 2 and system 3.	85
Figure 5.6 Achievable loss rate differentiation region under system 2.	86
Figure 5.7 Achievable loss rate differentiation region under DGCP.....	87
Figure 5.8 Loss rates under two systems.	88
Figure 5.9 The tail part of packets delay CDFs.	89
Figure 5.10 Total loss under system 1 and system 3.	90

List of Notations

Notations in EDF scheduler:

$A_i[\tau, \tau + t]$	The amount of class i traffic arrived between time τ and $\tau + t$.
$A_i^*(t)$	The bound of class i traffic at any time period with length t .
$S_i^{\max}$	The maximum transmission time of any packet in class i .

Notations in Fair Queueing scheduler:

S_i^k	The virtual start time of packet i of queue k .
F_i^k	The virtual finish time of packet i of queue k .
$W_i(\tau, t)$	The service queue I received during time period τ and t .
F^s	The fairness bound of scheduler s .
C_i^s	The latency of queue i in scheduler s .

Notations in RED:

$\min_{th}$	RED minimum threshold.
$\max_{th}$	RED maximum threshold.
$\max_p$	RED maximum drop probability.

Notations in DGCP:

$L_k(t_x), L_k^v(t_x)$	The virtual queue length of queue k when packet x arrives.
$W_k(t)$	The service queue k receives up to time t .
F_x	Estimated departure time of packet x .
R_x	Estimated departure round of packet x .
$R(t)$	The service round of a DRR/WRR server at time t .
$L(x)$	The length of packet x .
W_i^k	The service queue i receives in round k .
$W_{i,j}^k$	The bandwidth borrowed by queue i from queue j and used in round k

List of Acronyms

AF	Assured Forwarding
BE	Best Effort
CAC	Connection Admission Control
CBP	Complete Buffer Partitioning
DGCP	Delay Guaranteed Cross Pushout
DiffServ	Differentiated Services
DRR	Deficit Round Robin
DSCP	Differentiated Service Code Point
EDF	Early Deadline First
EF	Expedite Forwarding
FIFO	First In First Out
FQ	Fair Queueing
GPS	Generalized Processor Sharing
HOL	Head Of Line
LR	Latency Rate
MPLS	Multiple Protocol Label Switching
PBS	Partial Buffer Sharing
PGPS	Packet-by-packet Generalized Processor Sharing
PHB	Per Hop Behavior
PLR	Proportional Loss Rate
PQ	Priority Queueing
QM	Queue Management
QoS	Quality of Service
RED	Random Early Discard
RIO	RED with In and Out
SLA	Service Level Agreement
SP	Static Priority
WFQ	Weighted Fair Queueing
WRED	Weighted Random Early Discard
WRR	Weighted Round Robin
WTP	Waiting Time Proportional

Chapter 1

Introduction

1.1 Background

Traditional Internet that only provides best effort service cannot meet the diverse QoS requirements of today's Internet applications. DiffServ architecture [1] has been proposed as a scalable model to provide QoS in Internet. Three classes of services are defined in the DiffServ architecture: EF (Expedite Forwarding) [2], AF (Assured Forwarding) [3] and BE (Best Effort). EF service provides QoS similar to the leased virtual line; packets in EF service experience low delay, jitter and loss. AF service class has four subclasses: AF1~AF4, each can be used to guarantee certain QoS. BE service does not guarantee any QoS except for a minimum throughput.

In a DiffServ network, routers are categorized into edge routers and core routers. Edge routers condition the traffic from each customer to enforce the Service Level Agreements. Packets within the service profile are allowed to enter the DiffServ domain and treated according to their service classes. Packets out of the service profile are either dropped or marked with high drop precedence. In the DiffServ domain, flows with similar QoS requirements are mapped to the same service class and aggregated. To achieve good scalability, resources such as bandwidth and buffer space are assigned to each service class to provide service class based differentiated forwarding. For BE service class, only

a minimum bandwidth is reserved to prevent service starvation. For EF service class, resources are often over-provisioned to provide the superior service. Since the amount of EF traffic is usually only a small fraction of the total network traffic, over-provisioning will not affect the network utilization much. Unlike EF, resources required by AF service classes are allocated with a goal to achieve high network resource utilization. How to allocate resources depends on many factors: input traffic characteristics, QoS requirements and scheduling algorithms.

The most prevalent schedulers used today to support DiffServ AF service classes are based on fair queueing concept. Weighted Fair Queueing (WFQ) [4][5], Deficit Round Robin (DRR) [6] and Weighted Round Robin (WRR) [7] are some examples of fair queueing schedulers [4]-[17]. Under a fair queueing scheduler, all QoS requirements (delay, throughput, loss rate etc.) can be mapped to bandwidth requirement. Theoretically, given the QoS requirements and the input traffic characteristics, methods based on effective bandwidth or traffic envelope can be used to allocate bandwidth precisely. However, in reality, it is hard to predict the traffic in a core router of IP DiffServ network. The main reasons are: 1) Internet traffic is bursty. Even a single traffic source can generate long-range dependent traffic with analytically intractable traffic characteristics. 2) Although traffic is conditioned at edge routers, destinations of traffic flows are dynamic and random. As a result, traffic characteristics of flow aggregates at a core router change dynamically. Resources cannot be allocated precisely to flow aggregates in a static way.

One approach [18] to solve this problem is to dynamically assign bandwidth. However, the requirement of tracking traffic flows and modifying bandwidth allocations along their

routes introduces complexity to core routers. Typically a connection oriented network like MPLS (Multiple Protocol Label Switching) is required. In this thesis, we assume the environment of connectionless DiffServ IP networks where resource allocations are relatively static. Under static resource allocations, packet delays depend on buffer space, queue management scheme and the allocated bandwidth. Given the queue size and bandwidth allocation, the delay bound of packets in a queue can be calculated under a fair queueing system. Therefore, hard QoS in terms of packets delay bound can be guaranteed to each service class by allocating certain buffer space and bandwidth to it regardless of the traffic. The major problem with static resource allocations is that the loss of each class is dependent on the dynamic traffic and therefore subject to change.

However, although the total loss rate of all the service classes may not be improved due to the work conserving principle, it is possible to adjust the loss rates among individual service classes and provide soft QoS instead of hard QoS. By using the term soft QoS, we mean service classes with higher priorities get better services than service classes with lower priorities. Using loss rate as an example, the loss rate of a higher priority class is guaranteed to be smaller than the loss rate of a lower priority class. The proportional differentiated service model [28] proposed in [29] and [35] takes a further step to provide a controllable and predictable relative differentiated service. Let L_i , L_j be the average loss rates of class i and class j respectively, in proportional differentiated model, their loss rates satisfy the following equation: $L_i/L_j = \sigma_i/\sigma_j$. Where the σ_i are the loss rate differentiation parameters set by a service provider. By modifying σ_i , the quality spacing among the service classes can be adjusted.

The forwarding engine architecture in [35] contains two major elements: A WTP scheduler to provide proportional delay differentiation and a PLR dropper to achieve the proportional loss differentiation. QoS on packets delay and loss rates are both provided in terms of Soft QoS. However, in reality the end users often do not care or hardly notice the relative delay, it is the absolute delay or delay bounds that matters. For instance, in video on demand service, packets that miss their playback deadlines are useless no matter how good the relative service is. So, providing hard QoS on packets delay is of practical importance. On the other hand, unlike delay, most applications especially multimedia applications can tolerate certain loss. Using the same video on demand example, the playback quality declines gradually when loss rate increases. Service differentiation can be achieved effectively by providing soft QoS on loss rates.

The main objective of the research presented in this thesis is to develop a packet forwarding engine to provide hard QoS on delay and soft QoS on loss rates for DiffServ AF service classes. We choose Fair Queueing schedulers in our proposed forwarding engine because: 1) Fair queueing scheduler can provide hard QoS on packets delay. 2) Fair queueing schedulers are the most popular schedulers used in today's Internet to provide DiffServ AF services. A design based on fair queueing scheduler is compatible with the existing forwarding engines and therefore has better chance to be deployed.

We borrow the PLR dropper concept in [35] to provide loss rate differentiation. However, as we shall see later, applying PLR dropper directly to the fair queueing schedulers causes new problems. Although PLR dropper can control the loss rates, hard QoS on packet delays is destroyed. To solve the problem, a novel queue management scheme, the Delay Guaranteed Cross Pushout (DGCP) is proposed. With DGCP, the

proposed forwarding engine can not only guarantee the hard QoS on packet delays designated by the existing fair queueing based forwarding engine but also provides proportional differentiated service on packets loss.

1.2 Thesis Contributions

The contributions of the thesis are:

1. Develop a Delay Guaranteed Cross Pushout (DGCP) scheme to handle the cross class pushout under generic fair queueing schedulers. Unlike the typical pushout schemes that work within a FIFO queue, to our best knowledge, DGCP is the first pushout scheme that operates on different queues in a fair queueing system.
2. Prove that if the packets are of the same size, DGCP achieves two goals at the same time: 1) Packet delays of all classes are still bounded; 2) The total loss rate is fixed.
3. Generalize DGCP to handle packets with variable sizes under the Deficit Round Robin scheduler. With variable packet sizes, the first goal, providing delay bounds, is proven to be still achieved with the second goal, maintaining the same total loss, relaxing a little bit.
4. Based on DGCP, propose a packet forwarding engine under general fair queueing schedulers. Under static bandwidth allocation and dynamic traffic, the forwarding engine is able to provide hard QoS on delay and soft QoS on loss at the same time.

1.3 Thesis Organization

The remaining chapters of the thesis are organized as follows:

Chapter 2: Reviews DiffServ architecture and the related scheduling algorithms.

Chapter 3: Reviews the schemes in the literature for providing loss differentiations.

Introduces the proportional loss rates differentiation model.

Chapter 4: Introduces our proposed scheme: the Delay Guaranteed Cross Pushout (DGCP). Proves the delay bound guarantee and total loss guarantee under the general fair queueing schedulers with fixed packets sizes. Provides DGCP implementation under Weighted Fair Queueing Scheduler. In the last section, extends DGCP to DGCP+ to work under Deficit Round Robin scheduler with variable packets sizes and proves the delay bounds guarantee under DGCP+.

Chapter 5: Presents the simulation results. Verifies the delay bounds under DGCP and DGCP+ as well as the total loss guarantee under DGCP. Shows the total loss remains very close to the system without loss control under DGCP+.

Chapter 6: Presents conclusions and recommendations for future research.

Chapter 2

Differentiated Services

2.1 Introduction

Traditional IP networks only provide best effort service. All the packets enter the same buffer and are served in a FIFO mode. There is no attempt to provide service differentiation among flows. If the traffic load is far below the service capacity, this may not cause problem since all packets receive low delay and loss. However, due to the limited network resources (bandwidth, buffer space) and the increasing amount of traffic in the Internet, sporadic and periodically sustained congestions have been observed on current IP networks. Under congestion, packets experience longer delays and larger loss rates. QoS requirements of real time applications such as Voice over IP or Live Video cannot be satisfied by the best effort service. The requirement of providing more reliable service with certain Quality of Service (QoS) guarantees arises.

Differentiated Services (DiffServ) [1] has been proposed as a scalable model of providing QoS in Internet. In DiffServ, moving the intelligence to the edge routers and keeping the core routers simple bring good scalability. Edge routers are responsible for flow classification and traffic conditioning. Generally, flows with similar QoS requirements are mapped to the same service class. In the core routers, flows belonging to the same service class are aggregated and forwarded using the same Per Hop Behavior

(PHB). Service differentiation is provided by implementing different Per Hop Behaviors (PHB).

2.2 Core and Edge functions

The routers in a DiffServ domain are categorized into two classes. The edge routers, which are located at the edge of the DiffServ domain, classify the packets and condition the traffic. The core routers provide differentiated forwarding services by implementing different PHBs.

2.2.1 Edge Functions

Edge routers are the gateway routers to a DiffServ domain. In the edge routers packets are classified according to the Service Level Agreements (SLAs) between DiffServ domains and have their DSCP fields marked to specify the expected Per Hop Behaviors (PHBs). Traffic conditioning functions such as metering, traffic shaping or policing might be performed to ensure that the traffic conforms to the corresponded traffic profiles. Traffic shaping can change the input traffic characteristics by delaying some packets and force the input traffic to become adhere to the traffic profile. Compared to traffic shaping, in traffic policing, traffic is policed according to its traffic profile. The in-profile traffic is forwarded as usual. The out-profile packets are either dropped or marked with a different DSCP value. The service to the out-profile packets are different from the in-profile packets. For example, under congestion, the out-profile packets are discarded first.

2.2.2 Core Functions

Core routers provide differentiated forwarding services to packets according to their DSCP fields. No flow states are kept in the core routers. Packets with the same DSCP value are treated with the same Per Hop Behavior. The Per Hop Behaviors are provided under certain queueing disciplines such as Priority Queueing, Weighted Fair Queueing and Multi-class RED. By allocating network resources, each PHB can guarantee certain QoS. Compared to the edge routers, the functions in core routers are much simpler. The core router only needs to inspect the DSCP field of the packet and forward it using the corresponded PHB.

2.3 Per Hop Behaviors

The DSCP field in the IP head has six bits; therefore at most 64 PHBs can be defined. Currently, only three types of service classes are defined by the Internet Engineering Task Force (IETF).

2.3.1 Expedited Forwarding Class

The EF PHB [2] specifies that the departure rate of a class of traffic must equal or exceed a configured rate. The configured rate does not depend on the traffic dynamics. If the edge router guarantees the input traffic rate is smaller than the configured rate, the traffic will see no (or very small) queues. Since the queueing delay, jitter and loss are all due to the queues traffic experiences while transmitting the network, EF PHB guarantees

a low loss, low delay, low jitter and assured bandwidth service. A virtual leased line service can be provided.

Usually, Priority Queueing (PQ) [36] is used to provide the EF service by allowing the EF traffic to preempt other traffic unlimitedly. EF packets are given the highest priority and always served first. Priority Queueing can provide good QoS to EF class. However, the traffic of EF class must be constrained to prevent service starvation to other service classes.

Another way of implementing EF PHB is to use Fair Queueing. By guaranteeing the configured rate, the delay of a constrained traffic source is bounded. To ensure small delay and at the same time allow certain amount of traffic burst, bandwidth is often over-provisioned to EF traffic. Typically, EF PHB is used by the voice traffic that has stringent delay and jitter requirements but small bandwidth requirement. Since the amount of EF traffic is usually only a small fraction of the total traffic in the network, over-provisioning will not affect the network utilization much. Besides Fair Queueing, CBQ [19] can also be used to provide EF PHB by giving the EF queue priority up to the configured rate.

2.3.2 Assured Forwarding Class

The AF PHBs [3] guarantee the traffic is forwarded with high probability as long as the traffic does not exceed the subscribed information rate (profile). To make the full utilization of the network resources, the traffic is allowed to exceed the profile, however, the out-profile traffic is not forwarded with the same service guaranteed to the in-profile traffic. Whether the traffic exceeds the profile or not is measured in the edge routers. The

core routers forward both in-profile and out-profile packets in the same order as they arrive. However, under congestion, the out-profile packets are discarded first.

AF PHBs are used to offer different levels of forwarding assurance to meet different QoS requirements. Currently four AF classes are defined and each AF class is reserved certain forwarding resources (bandwidth and buffer space) to provide differentiated forwarding services. IP packets with similar QoS requirements are mapped into one AF service class and have their DSCP fields marked accordingly. Packets within one AF class can also be marked differently according to whether they belong to in-profile or out-profile traffic. Three drop precedence values are defined for each AF class. Out-profile packets with the highest drop precedence value are discarded first under congestion.

AF services are usually provided via Fair Queueing schedulers. The QoS requirements such as delay, jitter, loss and throughput can all be satisfied under Fair Queueing schedulers by allocating certain buffer space and bandwidth. Since the memory is relatively cheap today, buffer space is not a rare resource compared to bandwidth. However, although a large buffer is affordable, buffer size is often limited to a service class to control the congestion and packet delays. Compared to buffer space, bandwidth is the resource that has to be allocated with a goal to maximize the resource utilization.

In this thesis, we propose a mechanism to provide the proportional loss rate differentiation among the AF classes. Scheduling and queue management algorithms used to provide AF classes will be discussed in detail later.

2.3.3 Best Effort Class

As the name indicates, BE PHB does not guarantee any QoS. Only a minimum throughput is guaranteed for the BE traffic to prevent service starvation. It can be implemented using Fair Queueing schedulers by allocating a minimum bandwidth to the BE traffic.

2.4 Scheduling Algorithms

It is necessary to study the scheduling disciplines in more detail to understand how the PHBs are provided. In this section we review the scheduling algorithms that are most commonly used to provide DiffServ PHBs. We also discuss the connection admission control mechanism and the resource allocation schemes under some schedulers.

2.4.1 Static Priority (SP) Scheduler

In static priority scheduler [36], each queue is assigned a static priority. The scheduler always picks the head of line (HOL) packet from the queue with the highest priority to serve. As a result, packets belong to the queue with the highest priority get expedited forwarding service. SP scheduler is a natural choice to provide EF PHB. One significant problem with SP scheduler is low priority traffic has a longer delay and in an extreme case may experience service starvation. To prevent this, traffic with higher priorities has to be constrained. SP is not a good choice to provide AF PHBs, since there is no strict priority among different AF classes. Generally, SP is used to provide EF service by giving EF packets the high priority. Traffic from all other service classes is treated as low priority traffic.

2.4.2 Earliest Deadline First (EDF) Scheduler

EDF scheduler is usually used to provide delay bound guaranteed services. In EDF scheduler, each packet is assigned a deadline equals to the sum of its arrival time and the required delay bound. EDF scheduler then serves the packets in the increasing order of their deadlines. A well known advantage of EDF scheduler is its optimal efficiency in guarantee delay bounds: for a given set of connections with their associated delay bounds, EDF can provide delay guarantees that are at least as tight as any other type of schedulers [22]. In other words, if a combination of connection and their delay bounds can be satisfied under any type of scheduler, the same delay bounds are also guaranteed under EDF. If strict connect admission control is used, the number of connections admitted by EDF will be at least no less than any other type of schedulers.

Connection admission control functions given in [22] are based on traffic envelop with the assumption that the traffic is under the leaky bucket constraints. Here, we simply present the results in [22] and neglect the mathematical proof. Interested reader may refer to [22] for details.

Let $A_i[\tau, \tau + t]$ denote the total incoming traffic of class i between time τ and $\tau + t$. A traffic constraint function A_i^* provides a time invariant bounds on A_i if:

$$A_i^*(t) \geq A_i[\tau, \tau + t] \quad \forall t \geq 0, \forall \tau \geq 0 \quad (2.1)$$

Suppose the leaky bucket with token generation rate ρ and bucket depth σ is used to constrain the traffic, then the constraint function has the following format:

$$A_i^*(t) = \sigma + \rho * t \quad \forall t \geq 0 \quad (2.2)$$

Let d_i denote the delay bound of traffic class i , $d_j < d_k$ if $j < k$. Let $S_i^{\max}$ denote the maximum transmission time of any packet in class i , we have the following sufficient and necessary condition for EDF connection admission control:

$$t \geq \sum_i A_i^* (t - d_i) + \max_{d_j > t} S_j^{\max} \quad \text{for all } t \geq d_1 \quad (2.3)$$

Equation (2.3) shows the CAC of EDF scheduler under single node case. Given the traffic envelopes and delay bounds, it has a simple additive form. EDF scheduler can provide delay bound guarantees efficiently in a single node case. However, using the EDF schedulers to provide end-to-end service becomes more complicated. The main reason is that, in equation (2.3), we suppose the traffic is leaky bucket constrained. This is true at the edge routers where traffic shaping is performed. But when the traffic enters the DiffServ domain and is forwarded along its route, it may get distorted. The traffic can become more bursty and no longer under the leaky bucket constraints. As a result, the same delay bounds can no longer be guaranteed in the downstream nodes using equation (2.3). Furthermore, since there is no isolation among service classes, a bursty flow can affect the QoS of all other flows in the system. Therefore, it is difficult to provide end-to-end QoS [23]-[27] in a large network using EDF schedulers. One approach [23] to solve this problem and provide end-to-end delay guarantee under EDF is to add shapers in all routers along the path. However, this approach adds the complexity to the core routers and cannot be accepted under the DiffServ architecture.

2.4.3 Fair Queueing Schedulers

All fair queueing schedulers are based on the Generalized Processor Sharing (GPS) [4], [5] concept. Under fair queueing scheduler, each queue is guaranteed certain share of bandwidth. Compared to EDF, a major advantage of fair queueing schedulers is their ability to provide service isolation. In fair queueing schedulers, QoS of one queue will not be affected by service to other queues, and therefore can be guaranteed independently by reserving certain bandwidth to it. All QoS requirements can be mapped to bandwidth requirement under FQ Schedulers. Connection admission control becomes a simple query of whether there is enough bandwidth left. How to allocate the bandwidth under FQ scheduler to guarantee the QoS is beyond the scope of this research. Methods like effective bandwidth have been studied extensively and can be used to assign bandwidth. Another important property of fair queueing scheduler is that the end-to-end delay bound can be guaranteed with the shaper only applied at the edge of the network. With all these advantages, fair queueing schedulers become the most prominent schedulers used today to support AF classes. In this thesis, we will focus on the problem of providing AF services under fair queueing schedulers.

Fair queueing algorithms have attracted considerable research attentions in the past decade. A lot of scheduling algorithms have been proposed. We only discuss the GPS model and the fair queueing schedulers that are closely related to our design.

2.4.3.1 Generalized Processor Sharing (GPS) Multiplexing

GPS is based on fluid model where packet can be divided infinitesimally. Suppose the GPS server works at a fixed rate r . Let ϕ_i denote the weight assigned to queue i , g_i denote the service rate of queue i . Under GPS, we have:

$$\frac{g_i}{g_j} = \frac{\phi_i}{\phi_j}, \text{ for any queue } i \text{ and } j \text{ if they are backlogged} \quad (2.4)$$

Sum up equation (2.4) for all j , we have:

$$g_i = \frac{\phi_i}{\sum_j \phi_j} r, \text{ where } j \text{ belongs to the queues with backlogs} \quad (2.5)$$

Under the worst case, i.e. all queues have backlogs, we have:

$$g_i = \frac{\phi_i}{\sum \phi_j} r, \text{ where the summation is over all queues} \quad (2.6)$$

Equation (2.4) guarantees the fairness: at any time instance, the service rate of a queue is in proportional to its assigned weight. From equation (2.6), we can see that the queue is guaranteed a minimum service rate regardless of the status of other queues. Therefore, service isolation is provided among queues.

2.4.3.2 Packet-by-packet GPS (PGPS)

GPS is an ideal model that guarantees the absolute fairness among competing queues. It is based on fluid model. However, in reality, the transmission of a packet should not be interrupted. Packet-by-packet GPS is proposed as an emulation of GPS under more practical assumption that the server transmits one packet at a time.

PGPS uses virtual time to record the service completed under a parallel GPS system. When a packet arrives, its virtual finish time is computed as the finish time under the

GPS system. To emulate the GPS, the packets departure sequence under PGPS and GPS should be kept as similar as possible. If a packet finishes transmission earlier under GPS system, it should also be served earlier under PGPS. Therefore, PGPS server always selects the packet with the smallest virtual finish time to serve.

Suppose the k th packet of queue i arrives at (real) time a_i^k with length L_i^k , the virtual times that the packet starts and finishes service are calculated as following:

$$S_i^k = \max\{F_i^{k-1}, V(a_i^k)\} \quad (2.7)$$

$$F_i^k = S_i^k + \frac{L_i^k}{\phi_i} \quad (2.8)$$

From equation (2.7) and (2.8) we can see that the virtual finish time of a packet under PGPS is determined when the packet arrives at time a_i^k . Future arrivals will not change the virtual finish times of the already backlogged packets. As a result, the departure sequence of the backlogged packets does not depend on the future arrivals.

Updating the system virtual time in PGPS server is complicated; the virtual time increases at a speed inversely proportional to the summation of the weights of all backlogged queues. Since the number of backlogged queues can change at any time during a packet transmission time, in the worst case, PGPS has an $O(N)$ complexity. N is the number of queues.

2.4.3.3 Weighted Round Robin (WRR) Scheduler

Scheduling algorithms based on round robin concept have less complexity than PGPS. Under WRR [7], each queue is assigned an integer weight (quantum) Q_i . The server

serves each queue in a round robin manner. At each round, Q_i packets from queue i will be served. Assuming all the packets in the system have the same size, the WRR server aims to serve a queue at the rate of $\frac{Q_i}{\sum_j Q_j}$, where j belongs to the queues with backlogs.

The algorithm has an $O(I)$ complexity since the server serves the queue in turn. However, the delay of a packet can be very long. Suppose a packet just misses its scheduling turn, it has to wait until its next scheduling round. In the worst case, all the queues has enough backlogs, the packet may wait F time slots to be served. $F = \sum Q_i$ is the frame length.

2.4.3.4 Deficit Round Robin (DRR) Scheduler

WRR is designed to work under systems with fixed packet size. DRR improves WRR to handle packets with different sizes. In DRR [6], a quantum Q_i in terms of bytes per round is assigned to each queue. At the beginning, queue i is allowed to send Q_i bytes at the first round. Since packet sizes vary, the amount of bytes sent may be smaller than Q_i . The difference is recorded in deficit D_i . In the next round, $Q_i + D_i$ bytes are allowed to be sent. By keeping record of the deficit, fairness is guaranteed. DRR has the same problem with WRR. If a packet has just misses its round to be scheduled, it may wait until $F = \sum (Q_i + D_i)$ bytes are transmitted. As an algorithm based on round robin, DRR has $O(I)$ complexity.

2.4.3.5 Fairness of Fair Queueing Schedulers

In general, fair queueing schedulers can be categorized into two types. Schedulers of the first type are based on sorted priority. A packet is assigned a tag when it arrives. The scheduler then selects the packet to serve according to the tags. If the number of queues is N , the complexity of sorted priority based algorithm is at least $O(\log_2 N)$. Since the server selects the packet to serve after serving each packet, it usually has a closer emulation to GPS system. PGPS is an example of the sorted priority based server where virtual finish time is used as the tag. Schedulers of the second type are frame based schedulers. Schedulers belong to this category serve the queues in certain sequence and serve multiple packets from each queue at a time. WRR and DRR are examples of frame based schedulers. This type of scheduler has an $O(1)$ complexity but can only emulate the GPS coarsely.

One important property of fair queueing schedulers is their ability to guarantee the fairness. In other words, the service $W_i(\tau, t)$ to a queue i should be in proportional with its weight ϕ_i . The fairness F^s of server F is defined at following: At any period of time (τ, t) , the difference of normalized service $\frac{W_i(\tau, t)}{\phi_i}$ between any two continuously backlogged queues is within a bound:

$$\left| \frac{W_i(\tau, t)}{\phi_i} - \frac{W_j(\tau, t)}{\phi_j} \right| \leq F^s \quad (2.9)$$

The fairness F^s of GPS, PGPS, WRR and DRR is given in the following table:

Table 2.1 Fairness under Fair Queueing Schedulers.

Scheduler	Fairness
GPS	0
PGPS	$\max(C_j + \frac{L_{\max}}{\phi_i} + \frac{L_j}{\phi_j}, C_i + \frac{L_{\max}}{\phi_j} + \frac{L_i}{\phi_i}),$ Where $C_i = \min((N-1)\frac{L_{\max}}{\phi_i}, \max_{1 \leq n \leq N}(\frac{L_n}{\phi_n}))$
WRR	$\frac{F}{r}$
DRR	$\frac{3F}{r}$

Where L_i is the maximum packet size of queue i , $L_{\max}$ is the maximum packet size in the system respectively.

From Table 2.1, we can see that GPS is perfectly fair in that within any time period the difference of the normalized service received by two backlogged sessions are 0. For PGPS, the format of F^s is complicated. However, it is a function of individual packet lengths. For the frame-based scheduler such as DRR and WRR, F^s is a function of the frame length, which is typically much larger than the sum of a few packet lengths. Therefore, PGPS provides better fairness as a closer emulation to the GPS server.

2.4.3.6 Delay Bounds under Fair Queueing Schedulers

With the fairness guarantee and service isolation, it is possible for FQ schedulers to provide QoS guarantee by allocating certain bandwidth. To study the throughput and

delay guarantees under FQ, it is necessary to introduce the Latency-Rate (LR) server [9] concept. A LR server is defined as following:

Let τ be the starting time of the j th busy period of session i in server S and τ^* the time at which the last bit of traffic arrived during the j th busy period leaves the server. Then, server S is a LR server if and only if a nonnegative constant C_i^s can be found such that, at every time instant t in the interval (τ, τ^*) ,

$$W_i^s(\tau, t) \geq \max(0, \rho_i(t - \tau - C_i^s)) \quad (2.10)$$

Where ρ_i is the guaranteed rate to queue i , C_i^s is the latency that queue i can start service at the guaranteed rate.

An important result of LR server is that if the input traffic is regulated by a leaky bucket (σ_i, ρ_i) before it enters a network of LR servers, the worst case end-to-end delay of the packet following a path consists of k LR servers is bounded as following:

$$D_i \leq \frac{\sigma_i}{\rho_i} + \sum_{s=1}^k C_i^s \quad (2.11)$$

The end-to-end delay bound consists of two parts; the delay caused by the burstyness of input traffic and the latency experiences in each server along the path. Note that, the bound is provided with the traffic conditioned only once at the edge router of the network. Unlike using EDF scheduler, no shaper is needed in the core routers.

All the fair queueing schedulers discussed so far are LR servers. Table 2 shows their respective latency C_i^s .

Table 2.2 Latency of Fair Queueing Schedulers.

Scheduler	Latency
GPS	0
PGPS	$\frac{L_i}{\rho_i} + \frac{L_{\max}}{r}$
WRR	$\frac{F - Q_i L + L}{r}$
DRR	$\frac{3F - 2Q_i}{r}$

Where L is fixed packet size in the WRR scheduler.

From Table 2.2, we can see that as an ideal model GPS does not have any latency. A queue gets service immediately after its packet arrives. The latency of PGPS is a function of individual packet length. The latencies in both WRR and DRR are functions of the frame length. PGPS again provides lower latency as a result of a closer emulation of GPS.

2.5 QoS under Dynamic Traffic

With the latency known as listed in table 2.2, using equation (2.11), the worst case end-to-end delay of a leaky bucket constrained traffic source under network formed with fair queueing schedulers can be bounded. However, under a DiffServ network, things are more complicated.

First, bandwidth allocations are no longer for an individual flow. Flows aggregate and bifurcate in each router and bandwidth is assigned to flow aggregates. The interference of traffic flows makes the traffic characteristic of flow aggregates difficult to predict.

Second, each traffic source itself may generate long-range dependent traffic. For example, the traffic generated by a single video source is known to be self-similar. The traffic characteristics of such traffic are analytically intractable. Although the traffic is conditioned at the edge router, out-profile packets may still enter the network. The chance of congestion increases. The input traffic is no longer under the leaky-bucket constraints.

Third, the traffic conditioning at the edge is usually based on the SLA. Unless specified in the SLAs, the destinations of the traffic flows in an IP DiffServ network are dynamic and random. Furthermore, the paths under IP DiffServ network may change.

As a result, the traffic characteristics of flow aggregates in a core router change dynamically. Resources cannot be allocated precisely in a relatively static way. One policy [18] of solving the problem is to dynamically adjust the bandwidth allocations according to the traffic. However, the requirement of measuring the traffic and changing the bandwidth allocation on-line adds the complexity to the core router. The other way is to regulate the traffic through Traffic Engineering. Typically a connection oriented network like MPLS is required. In this research, we assume the environment of connectionless DiffServ IP networks where resource allocations are relatively static. The contradiction between dynamic traffic and static resource allocations is always there.

Under the static resource allocation, delay characteristics are dependent on the buffer space, queue management scheme and the allocated bandwidth. Delay for each packet varies, but in a fair queueing system with fixed buffer sizes and bandwidth allocations,

the worst-case delay for each queue is bounded. **The major problem with static resource allocation and the dynamic traffic is that the loss of each queue is dependent on the traffic and subject to change.** A mechanism that can adjust the loss rates according to certain QoS requirements is needed.

Chapter 3

Proportional Loss Rate

Differentiation Model

3.1 Introduction

As mentioned in chapter 2, in a DiffServ network, under relative static resources allocation and dynamic traffic, the loss rate of each service class is subject to change. However, although the total loss of all the service classes may not be improved due to the work conserving principle, it is possible to adjust the loss rates among service classes. Relative Differentiated Service model has been proposed to guarantee that service classes with higher priorities receive better service. The model has been refined to Proportional Differentiated Service [28]-[35] model to provide adjustable and consistent service differentiation among the service classes. In this thesis, we aim to provide delay guarantee for AF service classes and proportional loss rate differentiation among AF classes.

The rest of this chapter is organized as following: Section 3.2 reviews the queue management schemes in the literature to provide loss rate differentiation. Section 3.3

introduces the proportional loss rate differentiation model. Section 3.4 discusses the problem of providing loss rate differentiation under fair queueing schedulers.

3.2 Queue Management Schemes on Providing Loss Rate Differentiation

Prioritized buffer management schemes [37]-[45] have received considerable research attention. Numerous schemes have been proposed in the literature. In the following, we review the most significant buffer management schemes from the perspective of providing loss rate differentiation.

3.2.1 Pushout

In Pushout [37], [38] the buffer is totally shared by all service classes. Packet loss happens only when buffer overflows. A packet with higher priority can “pushout” a packet that is already in the buffer but with lower priority. According to which packet is to be dropped, Pushout algorithms can be further categorized into: First-In-First-Drop, Last-In-First-Drop or Random Pushout.

Pushout schemes do provide service differentiation in that the class with higher priority always receives better service, but it is difficult to adjust the quality spacing among classes. As an extreme case, the low service class may experience service starvation. Moreover, pushout schemes are proposed to adjust the loss rates within a FIFO buffer, loss rates among different FIFO queues served by the same scheduler cannot be handled by the existing Pushout schemes.

3.2.2 Partial Buffer Sharing (PBS)

In Partial Buffer Sharing (PBS) [36], only service class with the highest priority can use all the buffer space. Service classes with lower priorities can use the buffer only when the aggregate backlog is under certain thresholds. Service differentiation is provided by setting larger thresholds for classes with higher priorities. Compared to pushout schemes, the overall loss in PBS will be higher because the buffer is not fully utilized. The advantage of PBS is its simplicity in implementation since no Pushout is involved. A major problem with PBS is that the loss rates are highly sensitive to the traffic dynamics and thresholds and therefore cannot be finely tuned by changing the thresholds. With the static thresholds and the dynamic traffic load, the quality spacing among classes is difficult to control.

3.2.3 Complete Buffer Partitioning (CBP)

In CBP, the buffer space is completely partitioned to each service class. Packets from class i can only use the buffer space allocated to class i no matter if there is spare space in the buffer spaces allocated to other classes. According to Little's law, the backlogs of two service classes have the following ratio: $\frac{Q_i}{Q_j} = \frac{d_i \lambda_i}{d_j \lambda_j}$, where d_i is the average delay, λ_i is the average input rate of class i . The buffer should be partitioned according to the backlog ratio. CBP has the same problem as PBS. The loss rates are highly sensitive to parameter settings and traffic load. Under dynamic traffic, the service spacing can get distorted easily.

3.2.4 Multi-class RED

Random Early Discard (RED) [39] was proposed to work with TCP congestion control mechanism to improve the link utilization. TCP treats loss as the signal of congestion. In reaction of loss, TCP source halves its sending rate to avoid further congestion. However, if all TCP flows experience loss at the same time, the aggregate traffic rate decreases dramatically and causes under utilization of link capacity. This kind of problem is referred as TCP synchronization problem. RED avoids the synchronized loss by dropping the packets randomly and proactively before the buffer overflows. Two thresholds are set under RED. If the average queue length is below the $\min_{th}$ threshold, all the packets enter the buffer without loss. If the average queue length is above the $\max_{th}$ threshold, which indicates severe congestion, all the packets are dropped. The probability dropping happens in the congestion avoidance period when the average queue length is between the two thresholds. The dropping probability of a packet increases linearly from 0 to $\max_p$ as the average queue length grows from $\min_{th}$ to $\max_{th}$.

Weighted Random Early Discard (WRED) [40] was proposed to provide different loss precedence in a FIFO queue. In DiffServ AF class, packets are marked with different drop precedence. Higher priority packets have bigger thresholds $\min_{th}$ and $\max_{th}$. As a result, when the average queue length increases, the packets with higher drop precedence (lower priority) will be dropped first. WRED provide loss rates differentiation by setting different thresholds, therefore it is an algorithm based on PBS. Consequently, it can

provide loss rate differentiation but has difficulty in adjusting the quality spacing among different loss precedence.

RED with In and Out (RIO) [41] is another refined version of RED to provide loss rates differentiation. It is the same as WRED in setting different thresholds for packets with different loss precedence. The difference lies in the way of calculating the average queue length. In WRED, the average queue length is an exponential average of the queue length of all packets. In RIO, the average queue length of drop precedence i is the exponential average queue length of packets with drop precedence lower than i . The traffic formed by the packets with higher drop precedence will not affect the loss rate of packets with lower drop precedence. Therefore, in RIO, the packets with lower drop precedence are better protected. However, from the aspect of providing loss rate differentiation, as an algorithm based on PBS, no method of adjusting the quality spacing is provided by RIO.

3.3 Proportional Differentiated Service Model

The proportional differentiated service model states that the performance metrics (packets delay or loss rate) of service classes should be proportional to the service differentiation parameters set by the network administrator. The quality spacing can be controlled through the differentiation parameters that are set based on pricing or policy requirements. In particular, if the loss rate differentiation parameters are set to σ_i for class i , the loss rates should satisfy the following equation:

$$\frac{L_i}{L_j} = \frac{\sigma_i}{\sigma_j} \quad \text{where } L_i \text{ is the loss rate of class } i \quad (3.1)$$

The loss rate L_i of a class i is proportional to its corresponded loss rate differentiation parameter σ_i . The quality spacing among service classes can be adjusted by setting the loss rate differentiation parameters.

Besides the loss rate, proportional delay differentiation is proposed earlier in the proportional differentiated service model. The average delay d_i of each class i satisfies the following equation:

$$\frac{d_i}{d_j} = \frac{\delta_i}{\delta_j} \quad (3.2)$$

Where δ_i is the delay differentiation parameter for class i .

The architecture of proportional differentiated service model is shown in Fig. 3.1.

Figure 3.1 The packet-forwarding engine of proportional differentiated service model.

Packets are classified according to their service classes. Packets belong to the same class enter the same logical queue. All the per-class logical queues share the same physical buffer. Therefore, a loss happens only when the buffer overflows. The aggregate backlog controller monitors the buffer occupation and signals the loss to the proportional loss rate (PLR) dropper when the buffer overflows. It is the PLR dropper's task to

provide loss rate differentiation among the service classes. Upon receiving a drop signal, to satisfy equation (3.1), the PLR dropper drops a packet from the class with smallest normalized loss ratio $\frac{L_i}{\sigma_i}$. As a result, the proportional loss rate differentiation is achieved.

According to how the loss rate L_i is measured, the PLR dropper can be categorized into two classes. 1) PLR(∞). The loss rate is measured in long term for all the packets. 2) PLR(M). The loss is measured in short term. A sliding window with size M is used to count the loss and throughput. Whether the long term or the short term loss rate differentiation is preferred is an engineering issue.

For all the packets admitted, their delays are controlled by the proportional delay scheduler. Therefore, by pushing out another packet from a class with a smaller normalized loss ratio, the packet is guaranteed the relative delay according to its proportional delay parameter and gets its desired service.

PLR dropper works fine under the packet-forwarding engine of the proportional differentiated service model. However, the packet forwarding engine of a router that supports AF classes are different. Fig. 3.2 shows the forwarding engine used in the DiffServ model to support AF classes:

Figure 3.2 A general packet-forwarding engine for AF service classes.

The packet-forwarding engine in a typical DiffServ router uses the fair queueing scheduler. Unlike the proportional differentiated service model, the per-class logical

queues no longer share the same buffer space. Each logical queue has its own queue management scheme and buffer size allocation. A loss happens as the result of the local decision made by the QM of each logical queue. Both the delay and the loss rate of each service class are relatively independent to each other due to the service isolation under the fair queueing scheduler.

As discussed in chapter 2, under the relatively static allocation of bandwidth and buffer space, the loss is subject to change due to the dynamic traffic. A natural solution is to use PLR dropper to adjust the loss rate among the service classes. A forwarding engine with PLR dropper added is shown in Fig. 3.3.

Figure 3.3 System 2: A DiffServ packet forwarding engine with PLR dropper.

Each QM can signal the PLR dropper of a drop decision. Upon receiving the drop signal, like in the proportional differentiated service model, PLR dropper allows the packet to enter its queue if and only if there exists a packet in a class with a smaller normalized loss ratio. A packet in the class with the smallest normalized loss ratio is selected and dropped. By this way, the PLR dropper can provide the proportional loss rate differentiation. However, new problems arise.

We use a simple example to show what kind problems it may causes by simply adding the PLR dropper to a DiffServ packet-forwarding engine. Fig.3.4 shows a simple system with two service classes. The fair queueing scheduler used here is a WRR scheduler. It serves one packet from each class in each round. The queue management schemes for both queues are simply drop-tail. Packets are of the same size and both buffers have a size of N packets. The same loss rate is preferred for both service classes as the loss differentiation parameters for both service classes are set to the same value. If there is no PLR dropper the worst-case delay of a packet in the system is $2N$ times the packet transmission time.

Suppose at time t_0 , both queues are full. The scheduler is about to serve class 0. Class 1 has a smaller normalized loss ratio. When a new packet a of class 0 arrives, the PLR dropper pushes out a packet from queue 1 and admits a . After the cross pushout, the queue size of class 0 and 1 becomes $N+1$ and $N-1$ respectively. If there is no future arrival, a will be served within $2N$ transmission times after its arrival. The same delay bound as in a system without PLR dropper being provided. However, if a new packet b of class 1 arrives before packet a is served, the service of packet a will be delayed for one packet transmission time. Figure 3.4 shows the queueing dynamics for the above scenario. Although the loss for class 0 is avoided, packet a violates the delay bound. Due to the service isolation under fair queueing system, pushing out a packet from another class will not improve the service to the class triggering the pushout. Therefore, a longer delay is caused.

1. Both queues are full at the beginning. queue 1 has a smaller normalized loss ratio.

2. Packet *a* from queue 0 pushes out a packet from queue 1 and will be scheduled $2N$ transmission time slots after if there is no future arrivals.

3. After M rounds, a new packet *b* of class 1 arrives. The service to packet *a* is delayed. It will be served $2N+1$ transmission time slots after its arrival. A longer delay than the worst case delay without PLR dropper.

Figure 3.4 An example showing PLR dropper under fair queueing scheduler causes longer delay.

Using the same setting in the above example, suppose there is no future arrival of class 1 after packet *a* pushes out a packet from class 1 in t_0 . After one round, packet *c* of class 0 arrives. A new pushout will be triggered by packet *c* since queue size of class 0 reaches its limit of N . Two packets of class 1 are dropped. However, if there is no PLR dropper, packet *a* will be dropped but packet *c* will not. Only one packet is dropped. After applying the PLR dropper, the total loss becomes bigger. Fig.3.5 and Fig.3.6 show the overall loss under the two systems. The reason for a bigger overall loss is that after a pushout, the queue length of the queue that invokes the pushout increases. Because the service to the class is not necessarily improved by pushout, the queue length will stay in a bigger value for a longer time. The chance of future pushouts increases.

1. Both queues are full at the beginning. queue 1 has a smaller normalized loss ratio.

2. Packet *a* from queue 0 pushes out a packet from queue 1.

3. In the next round, a new packet *c* from queue 0 invokes another pushout. Altogether 2 pushouts are performed.

Figure 3.5 An example showing PLR dropper causes more overall loss under fair queueing scheduler.

1. Both queues are full at the beginning. queue 1 has a smaller normalized loss ratio.

2. Packet *a* of queue 0 is dropped if PLR dropper is not added

3. At the beginning of the next round. Both queues have *N-1* packets left. Packet *c* enters queue 0 directly. The overall loss under the system without PLR dropper is 1.

Figure 3.6 An example showing the overall loss under a system without PLR dropper.

From the above examples, we can see that the PLR dropper can provide loss rate differentiation under a fair queueing scheduler but can cause: 1) longer delay; 2) bigger overall loss. Delay violation may not be tolerable especially for those delay sensitive service classes. A bigger overall loss means that the service spacing is achieved at the cost of a worse overall service. A new scheme that can provide loss rate differentiation but avoids these costs is needed.

Chapter 4

Delay Guaranteed Cross Pushout

In this chapter, we propose a novel pushout scheme, the Delay Guaranteed Cross Pushout (DGCP). The scheme differs from pushout schemes in the literature in that: 1) Traditional pushout schemes only handle pushouts within a FIFO queue. To our best knowledge, DGCP is the first scheme to perform pushouts among different queues under a FQ scheduler. 2) Unlike other pushout schemes that push out packets based on priority, DGCP is delay aware and performs selective pushout. A new concept, Virtual Backlog is defined to ensure only those packets that satisfy certain delay requirements can be pushed out. 3) Different from the traditional pushout schemes in which a packet pushes out another packet to use the buffer space, in DGCP, a packet pushes out another packet to use both its buffer space and bandwidth.

The basic idea of DGCP is to associate service improvement with pushout by allowing bandwidth to be “borrowed”. DGCP is first proposed under the assumption that all packets have the same size like in the ATM system. Then it is generalized to work with variable packet sizes. The rest part of this chapter is organized as following: Section 1 describes the design principle of DGCP using a simple example and presents the conditions of performing the cross pushouts. Section 2 defines the virtual backlog used to meet the pushout conditions. Section 3 proves the most important property of DGCP, the

delay guarantee. Section 4 studies the overall loss under DGCP and proves that the total loss is fixed. Section 5 presents an implementation of DGCP under WRR scheduler and discusses its complexity. Section 6 generalizes DGCP to work under system with variable packet sizes.

4.1 Design Principles of DGCP

As shown in the previous examples in section 3.3, the cross pushout triggered by PLR dropper cannot improve the service to a class due to the service isolation under fair queueing scheduler. As a result, the loss rate differentiation is provided at the cost of more overall losses and longer delays. To solve the above problems, the cross pushout has to be related to service improvement. Therefore, by pushing out a packet from another class, the class triggering the pushout should not only get the chance of using the buffer space but also have the priority of using the bandwidth. Bandwidth should be “borrowed” after the pushout.

To allow bandwidth “borrowing”, the pushout scheme can be refined as follows: A packet α of class A can pushout a packet β from class B under certain circumstance. We will discuss the conditions to perform the cross pushout later. After the pushout, packet β is not dropped from class B ’s logical queue. Instead, it is marked as pushed out by class A . Later, when β ’s scheduling interval comes, the head of line packet from class A is scheduled and β is dropped. By doing so, the bandwidth assigned to β is actually used by class A . After pushing out packet β , because of the bandwidth “borrowing”, service to class A is improved.

We use a simple example to illustrate how service is improved. Consider a system with two service classes served by a round robin scheduler. The scheduler sends one packet from both service classes in each scheduling round. Both classes are assigned buffer size N and drop-tail is used as the QM scheme. At the beginning, both queues are full and the server is about to serve class 0. Class 1 has a smaller normalized loss ratio. A new packet a of class 0 marks packet b from class 1. When packet b 's scheduling interval comes, it is dropped and the HOL packet of class 0 is served. The backlog before a becomes $N-1$ after b 's scheduling interval. Any future arrivals of class 1 will not delay the service to packet a . Packet a is guaranteed to be scheduled after $2N$ packets transmission time. The same delay bound as under the fair queueing system without cross pushout is guaranteed.

Figure 4.1 shows the queueing dynamics under the proposed cross pushout scheme.

Figure 4.1 A example showing the bandwidth "borrowing" can guarantee the delay.

The above example shows that the delay bound is guaranteed due to bandwidth “borrowing”. It is necessary to ensure that packet b will be scheduled before a , otherwise, its scheduling interval cannot be used before a is served and service to packet a will not be improved. Different from the traditional pushout schemes, pushout cannot be performed against any packets in a queue. Only packets satisfy certain delay requirement can be pushed out. The pushout conditions should be refined as following:

A packet α of class A can pushout a packet β from class B if and only if:

- 1) Packet β will be scheduled before the tail packet (the packet queued before α) of class A if there are no future arrivals.**
- 2) Packet β is not marked as already been pushed out by another packet.**

As we shall see later, DGCP condition 1 guarantees packet α will meet its delay bound. DGCP condition 2 avoid “borrowing” the same bandwidth for multiple times.

4.2 Virtual Backlog

DGCP condition 1 is the key to guarantee service improvement. To meet condition 1, an effective method is needed to tell the departure sequence of two packets. As discussed in section 2.4.3, under fair queuing system without the bandwidth “borrowing”, the departure sequence of backlogged packets is not affected by future arrivals and can be determined when the packets arrive. Given the current backlog and bandwidth allocations, it is easy to find which packet will be served first.

Unfortunately, under DGCP, since bandwidth can be “borrowed” from each other, the backlog can no longer tell when a packet will be served. For example, a packet α may see

a backlog B_i before it, but among these B_i packets, K packets will be served using other queues' bandwidth. As a result, packet α will be served much earlier than the time needed for queue i to be served B_i times. Estimating the departure time based on current backlog is conservative since the packet may depart much earlier. Because of bandwidth "borrowing":

- 1) The departure sequence of two packets not only depends on the backlogs and the assigned bandwidths of both queues but also depends on the history of bandwidth "borrowing" among all queues. Estimating the depart time based on real backlog is too conservative. DGCP condition 1 cannot be checked based on real backlogs.
- 2) Queue management schemes based on real queue lengths are also too conservative and may cause unnecessary pushouts.

To solve the above problems, we define the so-called Virtual Backlog to estimate the packet departure time under DGCP:

Definition 1: The Virtual Backlog L_a of a queue A is the total amount of packets in the FQ system that are going to be served using queue A 's bandwidth.

Since we assume the packets sizes under DGCP are the same, the virtual backlog can be represented using number of packets. The way virtual backlog should be updated is shown in Fig. 4.2 and Fig. 4.3.

Upon packet arrival:

- 1) *if: a packet of class A enters its queue without pushout, increments virtual backlog of class A*
- 2) *else if: a packet of class A enters its queue by pushing out another packet from class B , virtual backlog of neither class A nor class B is updated.*
- 3) *else: the packet is dropped, no virtual backlog is updated*

Figure 4.2 Procedure to update virtual backlog upon packet arrival.

The virtual backlog is updated only when a packet is admitted by its queue management scheme and therefore will be transferred using its own bandwidth allocation. If a packet α enters queue A by pushing out another packet β of queue B , both queues' virtual backlogs are not updated. Although the real backlog of queue A increases by one, since bandwidth is borrowed, one more packet will be served using other queue's bandwidth, the total number of packets served using A 's bandwidth remains unchanged. For queue B , although packet β is pushed out, since β 's scheduling interval will be used by a packet from queue A , the number of packets that are going to be scheduled using B 's bandwidth

is not changed either. Therefore, the virtual backlogs of both queue A and B should be kept the same.

Figure 4.3 Procedure to update the virtual backlog upon packet departure.

Upon packet departure:

Scheduler selects a packet β from class B ; checks whether the packet is marked.

- 1) if not, sends the packet. Decrements the virtual backlog of class B .*
- 2) if marked as class X performed the pushout, drop the packet, checks the HOL packet of class X . The process goes on until an unmarked HOL packet is found.*

Sends the packet, decrements the virtual backlog of class B . Virtual backlogs of all other classes stay the same.

The above procedure shows how the virtual backlogs are updated upon a packet departure. Only the virtual backlog of the class whose scheduling interval is used gets decremented no matter packet from which queue is sent using this scheduling interval. Since the virtual backlog measures the number of packets using a class's bandwidth, when a packet departs, if it is using its own bandwidth, its virtual backlog decrements. If a packet from class A gets service using queue B 's bandwidth, A 's virtual backlog remains the same. It is B 's virtual backlog that decrements.

Let's use a simple example to show how virtual backlogs update. Consider a system with two service classes served by a round robin scheduler. The scheduler sends one packet from both service classes in each scheduling round. Both classes are assigned a logical queue with buffer size N . Drop-tail is used as the queue management scheme.

At the very beginning, both buffers are vacant, no cross pushout happens. At time t_0 , both queues are full and the server is serving a packet of class 0 . Class 1 has a smaller normalized loss ratio. It is supposed that the first cross pushout happens at time t_0 . Before

$t0$, since no cross pushout happened, the virtual backlogs update following the same rule as the real queue lengths. Therefore, both queues have a virtual backlog of N . At time $t0$, packet a of class 0 pushes out packet b from class 1 . The real queue lengths of queue 0 and queue 1 become $N+1$ and N respectively. After the cross pushout, no virtual backlogs are updated. Both virtual backlogs remain N . For queue 0 , only N of its $N+1$ packets are going to be served using its own bandwidth. For queue 1 , $N-1$ of its own packets and 1 packet from queue 0 will be served using its bandwidth. As the definition of virtual backlog, the virtual backlogs N of both queues reflect the number of packets that are going to be sent using the queue's bandwidth.

Later, when packet b 's scheduling interval comes, it is dropped and the HOL packet of class 0 is served. The real backlogs of queue 0 and queue 1 become N and $N-1$ respectively. Since queue 1 's scheduling interval is used, although a packet from queue 0 is sent, it is the virtual backlog of queue 1 that decrements instead of queue 0 's. As a result, the virtual backlogs of queue 0 and queue 1 become N and $N-1$ respectively. The virtual backlogs are now become the same as their corresponding real queue lengths since no packet in the system is marked and all the packets in a queue will use the queue's own bandwidth to be scheduled. Figure 4.4 shows the queueing dynamics in the above example.

Figure 4.4 An example showing how the virtual backlogs are updated.

4.3 Delay Guarantee under DGCP

Packet delay in a fair queueing system depends on a lot of factors: the scheduling algorithm, the allocated bandwidth and the backlogs in the buffers. By carefully engineering the scheduler and queue management parameters, packet delay of each service class can be controlled to meet certain QoS requirements. Here, we are not interested in how to allocate the bandwidth and buffer spaces or what the preferred delays should be. It is the network administrators' responsibility to configure the parameters and control the delays. Since the objective of our design is to provide a loss control scheme that works smoothly with the existing fair queueing based forwarding engine; the delays should be kept as similar as possible to the original system. Recall system 2 in Fig.3.3, it is not a good solution in providing proportional loss rates differentiation because it causes longer delays. DGCP is proposed especially to solve this problem. With the bandwidth

“borrowing” mechanism, we expect the same delays can be maintained with DGCP. We compare the packet delays of two parallel systems:

System 1: The original system is a fair queueing based packet forwarding engine as shown in Fig.3.2. No cross class loss controller exists. If when a packet arrives, it is the L th packet of its logical queue, it will be served after the queue is scheduled L times by the FQ scheduler. In other words, it will depart at the queue’s L th scheduling interval after its arrival. The queueing delay for this packet depends on the backlogs of other queues. If all the other queues are vacant, this queue will be served at full link speed and its L th scheduling interval comes sooner. Under the worst case, if all the other queues always have enough backlogs, the queue can only be served at its assigned bandwidth. The packet’s worst-case delay is bounded and independent to the backlogs of other queues due to the service isolation under fair queueing scheduler. The bound can be calculated given the maximum backlog allowed by the queue’s QM, the queue’s bandwidth allocation and the latency of the FQ scheduler.

Figure 4.5 System 3: A DiffServ packet forwarding engine with DGCP.

System 3: The forwarding engine is shown in Fig.4.5. We name it as System 3 since System 2 has already been defined in section 3.3 (Figure 3.3) to denote the DiffServ

forwarding engine with PLR dropper. The scheduler type, scheduler parameters (weights under PGPS, quantum under WRR etc.), queue management schemes and parameters are inherited from system 1. The extra parts are the PLR dropper and DGCP. The two work together to control loss rates among classes. PLR dropper keeps track of the loss rates of each class and selects the queue to perform the pushout with the target to achieve proportional loss rate differentiation. DGCP checks the pushout conditions to ensure the delay guarantee and marks the pushed out packet.

The scheduler serves each class/queue according to the fair queueing concept. The only difference is that it supports the bandwidth “borrowing”. Marked packet will be dropped and the HOL packet from the queue invoking the pushout is served instead. Here it is necessary to clarify the difference between “service to the packets in the queue” and “service to the queue”. The former counts how many packets of certain queue are served and its service time is related to “bandwidth borrowing”. The latter is counted as how many times the scheduler serves the queue regardless of packets from which queues are actually served. Compared to “service to the packets in the queue”, the service time for “service to the queue” is much easier to predict since it is only related to the bandwidth a queue is assigned under the FQ scheduler and has nothing to do with the dynamic “bandwidth borrowing”. Since both system 1 and 3 have the same bandwidth allocations the worst case waiting time for a queue (not the packets in the queue) to be served L times under both systems are the same.

QMs under DGCP are based on virtual backlogs. QM parameters are unchanged from system 1. Therefore, if in system 1, a packet is admitted under the real queue length L , the packet will also be admitted in system 3 when the virtual backlog is L . In system 1, the

packet will be scheduled after the queue is served L times. Delay in system 3 is more complicated due to the “bandwidth borrowing”. As defined in the previous section, the virtual backlog records how many packets in the system will be scheduled using the queue’s bandwidth (service to the queue). If the virtual backlog of queue A is L_a , all the packets using queue A ’s bandwidth will be sent no later than queue A is served L_a times. Therefore, with virtual backlog, the delay of a packet can be related to the “service to the queue”. Since the worst case waiting time for “service to the queue” under both systems are the same, by controlling the virtual backlog $L_a=L$, the same delay bounds can be provided for packets that use queue A ’s bandwidth in system 3 and the packets of “queue” A in system 1.

There is still a gap before we can claim the same delay bound for “class” A packets can be guaranteed in both systems, because in system 3, packets of “class” A may use other queue’s bandwidth to be scheduled. However, the above discussion does give us some insight on how to control the packet delays using the virtual backlogs. The detailed proof of the delay bound guarantee will be carried out in the following sections. Behind the complicated logics, the basic reason for DGCP to guarantee the delay is simple: due to the “bandwidth borrowing”, pushout improves the service to a packet.

We now began the proof.

Theorem 1: Under DGCP, the tail packet of a queue K will be scheduled using K ’s scheduling interval. If there is no future arrival, the tail packet departs at the $L_k(t)$ th scheduling interval of queue K after its arrival, where $L_k(t)$ is the virtual backlog of queue K when the packet arrives at time t .

Proof: We shall prove the theorem in three steps.

Step 1: It is easy to see that the virtual backlog $L_k(t)$ of class K is a step function of time t . Assume the first cross pushout happens at time t_0 . When $t < t_0$, $L_k(t)$ equals to the real queue length of class K . As in system 1, which is a fair queueing system, the tail packet departs at the $L_k(t)$ th scheduling interval of queue K . Theorem 1 holds.

Step 2: At time t_0 , a packet from class A pushes out a packet from class B . Three cases may happen:

- 1) $A=K$. The new packet now becomes K 's tail packet and pushed out packet β from class B . Since condition 1 of DGCP guarantees that packet β will be scheduled before K 's tail packet, one extra packet from class A can be served using β 's time slot before K 's $L_k(t_0^-)$ th scheduling interval. As a result, the newly arriving packet, which is now the tail packet of K , will be served at $L_k(t_0^-)$ th scheduling interval *instead of* $L_k(t_0^-) + 1$ th scheduling interval. Since its virtual backlog is not updated after cross pushout as stated in the last section, $L_k(t_0) = L_k(t_0^-)$.
- 2) $B=K$. One packet α in K is pushed out. Since condition 1 in DGCP guarantees that the tail packet of A will be scheduled after α . when α 's scheduling interval comes, there exists a packet from class A to be transferred using α 's time slot. As a result, all the other packets except α in class K still get served at their original schedule intervals. The tail packet in K is served (dropped, if it is α) at $L_k(t_0^-)$ th scheduling interval. Since the virtual backlog of queue K is not updated after cross pushout as stated in the last section, $L_k(t_0) = L_k(t_0^-)$. Theorem 1 holds.
- 3) K is neither A nor B . The service to all the packets in class K is unchanged.

In summary of the above three scenarios, when $t=t_0$, the theorem holds.

Step 3: Suppose the next packet comes at time $t1$. Four scenarios may happen.

- 1) The packet enters class K directly. Because the original tail packet of class K is going to be scheduled at $L_k(t1^-)$ th scheduling interval of K , the new tail packet will be scheduled at $L_k(t1^-)+1$ th interval of K . Since $L_k(t1) = L_k(t1^-)+1$, theorem 1 holds.
- 2) One of the three scenarios in the last case may happen again. The same arguments apply.

Following the same logic in step 3, assuming theorem 1 holds after the i th incoming packet, theorem 1 will still hold after the $i+1$ th incoming packet. With the theorem holds at step 1 and 2, it will hold all the time based on proof by mathematical induction. Proof is complete.

Theorem 1 shows that, if there is no future arrival, the tail packet of queue K will be served after L_k scheduling interval of queue K . Since the maximum allowable queue lengths under system 1 and system 3 are set to the same, without future arrival, the tail packets under both systems will have the same worst-case delay. Corollary 1 studies the case when future arrivals are considered.

Corollary 1: A packet α of queue K is guaranteed to be served no later than the $L_k(t_\alpha)$ th scheduling interval of queue K after its arrival. Where t_α is the time α arrives; $L_k(t_\alpha)$ is the virtual backlog right after α arrives.

Proof: When packet α of class K arrives, it is the tail packet of K . If no future arrivals, from theorem 1, the tail packet will depart at the $L_k(t_\alpha)$ th scheduling interval of K . Corollary 1 holds.

Considering the future arrivals, the departure scheduling intervals of the already backlogged packets of class K may alter under four cases:

- 1) Packet γ queued before α in queue K is pushed out. DGCP guarantees that γ 's scheduling time will be used by a packet from the queue that invoked the pushout. As a result, packet α departs at the same scheduling interval. Corollary 1 holds.
- 2) Packet γ queued after α in queue K is pushed out. Nothing will happen to the scheduling interval of α .
- 3) Newly arrival packet of K pushes out another packet β that will be scheduled before α . Then, including α , the packets in K with scheduling interval after β will get service earlier. Packet α will depart before the $L_k(t_\alpha)$ th scheduling interval of K after its arrival. Corollary 1 holds.
- 4) Newly arrival packet of K pushes out another packet β that will be scheduled after α , nothing will happen to the scheduling interval of α .

Combine all cases, proof is complete.

Corollary 1 tells us that, when a packet is admitted, it is expected to be scheduled after $L_k(t_\alpha)$ th scheduling interval of its queue. The future arrivals will not delay the departure of the already backlogged packets. In fact, a packet may be served earlier due to the bandwidth “borrowed” from future pushouts.

With theorem 1 and corollary 1, we can now compare the delay under system 1 and system 3. If under system 1, a packet α is admitted by its QM when its real queue length is $L_k(t_\alpha)$, it will be scheduled after its queue is served $L_k(t_\alpha)$ times. Under system 3, admitting the packet when the virtual backlog is also $L_k(t_\alpha)$ guarantees the packet will

be scheduled no later than the queue is served $L_k(t_\alpha)$ times. As a result, we have the important property of DGCP:

Property 1: DGCP guarantees the same delay bounds as in the original fair queuing system if the QMs are based on virtual backlogs. All the packets admitted by DGCP will not violate the designated delay bounds.

4.4 Total Loss under DGCP

In the previous section, we have proven that in DGCP one can adjust the loss rates among service classes and all the packets are still guaranteed the delay bounds. In other words, the hard QoS on packet delay designated by the original packet-forwarding engine is kept and new function of providing soft QoS on packet loss is added. A natural question is whether the total loss is increased or decreased by performing the cross pushouts. We will answer the question in this section.

The total loss of a fair queueing system depends on a lot of factors: the bandwidth allocations, the queue management algorithms and the traffic dynamics. Under the static bandwidth and QM parameter settings, the total loss depends on the traffic dynamics and is subject to change. Here we are not interested in the absolute amount of the total loss, since it is up to the network administrator to set up the scheduler and QM parameters to control the overall loss. Furthermore, certain amount of loss is unavoidable due to the limited service capacity and service conserving discipline. Since the goal of our design is to provide loss rate differentiation it would be considered ideal if the goal is achieved without introducing any more total losses to the original forwarding engine.

As in the previous section in which the delay bounds are compared, we compare the total loss in this section under the same parallel systems. System 1 shown in Fig.3.2 is the original fair queueing system without any cross class loss rates control. System 3 shown in Fig.4.5 adds PLR dropper and DGCP to system 1. In order to study the changes of total loss due to DGCP, the scheduler and QM parameters and the input traffic are kept the same in both systems.

One may take for granted that the cross pushout will not affect the total loss since each time there will be exactly one packet being dropped no matter the cross pushout is successful or not. If all the queues share the same buffer size and the scheduler is work conserving, the total loss is a function of the aggregate input traffic and the output link speed and therefore is fixed no matter whether or how the pushouts are performed. However, in a system with complete buffer partitioning, each logical queue has its own buffer allocation and QM scheme, the work conserving feature of the scheduler alone can no longer guarantee the same total loss if cross pushout is allowed. Recall the examples in Fig.3.5 and Fig.3.6, the cross pushout in system 2 (shown in Fig.3.3, system 1 with PLR dropper but without DGCP) does cause more losses under the same input traffic. So comparing the total loss under system 1 and system 3 is not as simple as it looks. We need to take a closer look at both systems.

In system 1, the drop decisions are made by a logical queue's QM based on the queue's real queue length. The number of total losses in system 1 equals to the sum of the number of dropping decisions made by each logical queue's QM.

In system 3, a drop decisions are also made by the logical queue's QM but based on the queue's virtual backlog. Once a drop decision is made in system 3, DGCP may cross

pushout another packet in another queue or drop the packet if the cross pushout fails. In either case, there is one packet loss. Therefore, once a logical queue's QM makes a drop decision, there will be exactly one loss. Again, the number of total losses under system 3 again is the sum of the number of dropping decisions made by each logical queue's QM.

The above analysis shows that the dynamics of the real queue lengths and the virtual backlogs determine the total loss under system1 and system 3 respectively. Let's denote the real queue length of class K in system 1 as $L_{1k}(t)$, the virtual backlog of class K in system 3 as $L_{3k}(t)$. Both $L_{1k}(t)$ and $L_{3k}(t)$ are stochastic processes of time t . At any particular time t' , both processes have integer values that indicate the number of packets. A realization of $L_{1k}(t)$ or $L_{3k}(t)$ is a step function of time t and may look like the example shown in Fig.4.6.

With the $L_{1k}(t)$ and $L_{3k}(t)$ defined, we can prove the two systems have the same total loss. The surprising result comes from a simple reason: since the virtual backlogs are not updated after a cross pushout, the real queue length in system 1 and virtual backlog in system 3 are always synchronized. The following theorem states the above fact.

Figure 4.6 A realization of real queue length $L_{1k}(t)$ or virtual backlog $L_{3k}(t)$.

Theorem 2: Consider two parallel systems with the same input traffic and scheduler and QMs parameters. System 1 is a fair queuing system without cross

pushout and its real queue length of a queue K is $L_{1k}(t)$. System 3 is the system with DGCP and its virtual backlog of queue K is $L_{3k}(t)$. We have $L_{1k}(t) = L_{3k}(t)$ for all K, t .

Proof: We shall prove theorem 2 in four steps.

Step 1: At the beginning, the logical queues under both systems are vacant. Suppose the first cross pushout in system 3 happens at t_0 , before t_0 , both systems run on real queue lengths. Therefore, before t_0 , we have: $L_{1k}(t) = L_{3k}(t)$ for all K, t . Theorem 2 holds.

Step 2: Consider the first cross pushout in system 3. At time t_0 , a packet from queue A pushes out a packet from queue B . $L_{3a}(t_0)$ and $L_{3b}(t_0)$ are not updated since no virtual backlogs are updated after a cross pushout as stated in the last section. Therefore, $L_{1k}(t_0) = L_{3k}(t_0)$ for all K at time t_0 . Assume no future arrivals, according to theorem 1, the tail packet of queue K will be scheduled after $L_{3k}(t_0)$ scheduling intervals of K . Since all the $L_{3k}(t_0)$ still equal to $L_{1k}(t_0)$, the backlog of each queue will be cleared at the same scheduling interval of queue K under both system 1 and system 3. Since under both systems, the queue lengths (real or virtual) of any queue K decrement at K 's own scheduling intervals and all the backlogs clear at the same scheduling intervals, $L_{3k}(t) = L_{1k}(t)$ at any time instance after t_0 . Here, we have proven. $L_{1k}(t) = L_{3k}(t)$ for all K, t if after the first cross pushout, no future packet arrives.

Step 3: Next, suppose at time t_1 , a new packet of class C comes. We have $L_{3k}(t_1^-) = L_{1k}(t_1^-)$ for any class K as the result of step 1. Three cases may happen.

- 1) If the new packet of class C is accepted by one system, it will be accepted by both systems since $L_{3c}(t_1^-) = L_{1c}(t_1^-)$. Both $L_{3c}(t_1)$ and $L_{1c}(t_1)$ increments. As a result, $L_{3k}(t_1) = L_{1k}(t_1)$ for all K .

- 2) If the packet is dropped under both systems, nothing changes. $L_{3k}(t1) = L_{1k}(t1)$ for all K .
- 3) If the packet is dropped in system 1 but accepted by system 3 by cross pushing out another packet in queue D , the virtual backlogs in system 2 is not changed after cross pushout. $L_{3k}(t1) = L_{1k}(t1)$ for all K .

As a result, under each case, $L_{3k}(t1) = L_{1k}(t1)$ for all K . Again, assuming no future arrivals, according to theorem 1, the backlog of a queue K under system 3 will be cleared after $L_{3k}(t1)$ scheduling intervals of K . Since under both systems, the queue lengths (real or virtual) decrement at their own scheduling intervals and all the backlogs clear at the same scheduling interval, $L_{1k}(t) = L_{3k}(t)$ after $t1$, if there is no future arrival.

Step 4: If we partition the time domain after $t1$ as $[t1, t2]$, $[t2, t3]$, $[t3, t4]$, ..., $[ti-1, ti]$, ..., where ti is the time the i th incoming packet arrives, then we have proven $L_{1k}(t) = L_{3k}(t)$ for each inter arrival time period. As a result, $L_{1k}(t) = L_{3k}(t)$ at any time t .

Proof is complete.

Since the losses under both systems are completely determined by $L_{1k}(t)$ or $L_{3k}(t)$, we have the important property of DGCP:

Property 2: If the input traffic does not depend on loss, DGCP guarantees the same total loss as if no cross pushout is performed.

4.5 Departure Sequence under DGCP

In the previous two sections, we have shown that DGCP can guarantee the same delay bounds and total loss as in the original packet forwarding engine. The proofs of theorem 1 and 2 do not depend on the PLR dropper or the QMs. The only requirement is that

DGCP conditions are met while performing the cross pushouts. In this section, we will discuss how DGCP conditions can be met under a generic fair queueing system.

Here it is necessary to recall the DGCP conditions that are given in section 4.1:

A packet α of class A can pushout a packet β from class B if and only if:

- 1) Packet β will be scheduled before the tail packet (the packet queued before α) of class A if there is no future arrivals.**
- 2) Packet β is not marked as already been pushed out by another packet.**

Condition 2 can be checked by simply looking up a field in packet β to see if it has been marked already. The difficulty lies in meeting condition 1 which involves comparing the departure sequence of two packets.

Let's first look at how condition 1 can be met if there is no cross pushout. Generally, the method of telling the departure sequence of two packets is scheduler type dependent. For example, under PGPS, the packet with a smaller virtual finish time will be sent first. Under WRR, the departure sequence can be calculated using the queue lengths. Here, we derive a generic method that can be used under any type of fair queueing schedulers to determine the departure sequence of two packets.

Under fair queueing scheduler, packet a of queue A is guaranteed to be scheduled before packet b of queue B if:

$$\frac{L_a}{r_a} > \frac{L_b}{r_b} + F^s \quad (4.1)$$

Where L_a is the amount of backlog in number of packets before and including packet a in queue A . L_b is the amount of backlog in packets before and including packet b in queue B . r_a and r_b are the bandwidth allocations in packets per second to queue A and B

respectively. F^s is the fairness bound of the scheduler. Equation (4.1) is a direct result from equation (2.3) according to the definition of fairness bound (section 2.4.2). We now have a sufficient condition to ensure packet a will be scheduled after packet b given both queues' backlogs and bandwidth allocations.

However, under DGCP, due to the bandwidth borrowing among the queues, the backlogs and the bandwidth allocations can no longer determine the packet delay. Equation (4.1) needs to be refined using the virtual backlogs. We have:

Theorem 3: Each logical queue K records its progress of service $W_k(t)$. $W_k(t)$ is set to 0 at the beginning of a session backlog period and increments everytime when the queue is served. If a packet x of K arrives at time t_x , its estimated service finish time F_x is recorded as $F_x = W_k(t_x) + L_k(t_x)$, where $L_k(t_x)$ is the virtual backlog of K . DGCP condition 1 can be met if: A packet α of class A pushout a packet β from class B with the following inequation holds:

$$\frac{L_a(t_\alpha)}{r_a} > \frac{F_\beta - W_b(t_\alpha)}{r_b} + F^s \quad (4.2)$$

Proof: We will prove the theorem in three steps.

Step 1: At the beginning of a system backlog period, all the queues are vacant. No cross pushout happens. The virtual backlog of each queue equals to the real queue length. Assume the first cross pushout happens at time $t0$. At $t0$, equation (4.2) is checked using the real queue lengths. $L_a(t0)$ is the real queue length of queue A . $F_\beta - W_b(t0)$ is the length of backlog before and including packet β in queue B . As a result, equation (4.2) is actually equation (4.1). If there is no future arrivals, equation (4.1) guarantees that the tail packet before packet α will be scheduled before packet β . DGCP condition 1 holds.

Step 2: Suppose the second cross pushout happens at time $t1$. Packet γ of class C pushes out packet δ of class D . Since DGCP condition 1 holds before $t1$, Theorem 1 and Corollary 1 also hold. According to theorem 1, the tail packet before packet γ is to be scheduled after the queue C is served $L_c(t1)$ times. As a result of corollary 1, packet δ will be served no later than queue D is served $F_\delta - W_d(t1)$ times. According to the fairness bound, if there is no future arrival, the tail packet before γ will be served before δ . DGCP condition 1 holds.

Step 3: Following the same logic used in step 2, if the third cross pushout happens at time $t2$, DGCP condition 1 will still hold if equation (4.2) is checked to perform the pushout. Therefore, given theorem 3 holds at the i th cross pushout, theorem 3 will still hold at the $i+1$ th cross pushout. Since in step 1 we have already proven Theorem 3 holds at the first cross pushout, by induction, Theorem 3 holds at all cross pushouts. Proof is complete.

With the scheduler specific fairness bound F^δ , equation (4.2) is applicable to any type of fair queueing schedulers. In equation (4.2), the fairness bound F^δ and allocated bandwidth r_k are fixed; the virtual backlog L_k and the service W_k should be recorded for each queue. Keeping track of L_k and W_k is of $O(1)$ complexity since they only update when “service to the queue” happens. Theorem 3 provides a simple method to check DGCP condition 1, therefore, implement DGCP, under a generic fair queueing scheduler.

4.6 DGCP Implementation under WRR

Theorem 3 makes it possible to implement DGCP under a generic fair queueing scheduler. In this section, we will show how DGCP can be implemented under a specific scheduler. We use Weighted Round Robin scheduler as an example.

Let's recall how WRR scheduler (see section 2.4.4.3) works. Under WRR, each queue is assigned an integer weight (quantum) Q_i . The server serves each queue in a round robin manner. At each round, Q_i packets from queue i will be served if there is enough

backlog. The fairness bound of WRR server has a simple format: $F^s = \frac{F}{r}$. Where F is the frame length in packets, r is the link speed in terms of packets per second.

Substitute F^s in equation (4.2) with $\frac{F}{r}$, we can get the condition to perform DGCP under WRR scheduler:

$$\frac{L_a(t_\alpha)}{r_a} > \frac{F_\beta - W_b(t_\alpha)}{r_b} + \frac{F}{r} \quad (4.3)$$

Divide both sides of equation (4.3) by frame transmission time: $\frac{F}{r}$, we have:

$$\frac{L_a(t_\alpha)}{Q_a} > \frac{F_\beta - W_b(t_\alpha)}{Q_b} + 1 \quad (4.4)$$

Let's take a closer look at equation (4.4). The left hand side of equation (19), $\frac{L_a(t_\alpha)}{Q_a}$, is the number of rounds it takes after time t_α for the tail packet of A to be served. Similarly, $\frac{F_\beta - W_b(t_\alpha)}{Q_b}$ indicates the number of rounds needed after time t_α for packet β to be

served. The “+1” in the right hand side guarantees that packet β will be served before the tail packet of A regardless of which queue the WRR scheduler is serving at time t_α .

In equation (4.4), it is supposed that the virtual backlog L_k and the service W_k are maintained by each logical queue K . As a round robin scheduler, the WRR scheduler itself should be able to record the rounds $R(t)$ in each system busy period. It is unnecessary to maintain the service W_k for each logical queue separately. With the service round $R(t)$ maintained by the WRR scheduler, we have:

Theorem 4: Under Weighted Round Robin scheduler, a packet x records its estimated departure round as: $\frac{L_k(t_x)}{Q_k} + R(t_x)$. Where $L_k(t_x)$ is the virtual backlog of class K . $R(t_x)$ is the scheduler’s scheduling round. t_x is the time packet x arrives. Q_k is the quantum of class K . DGCP condition 1 holds if a new packet of class A pushes out packet β from class B with the following equation holds:

$$\frac{L_a(t_\alpha)}{Q_a} + R(t_\alpha) > \frac{L_b(t_\beta)}{Q_b} + R(t_\beta) + 1 \quad (4.5)$$

Where α is the tail packet of class A .

Proof: The proof of theorem 4 is similar to the proof of theorem 3. Actually theorem 4 is a special case of theorem 3 under a specific type of scheduler. The proof has 3 steps.

Step 1: At the beginning of a busy period, all the buffers are vacant, no cross pushout happens. Assume the first cross pushout happens at t_0 , the virtual backlog of all queues before t_0 equals to the real queue length. For the first pushout, equation (4.5) is checked under real queue length. In WRR scheduler, $Q_k \geq 1$ packets are scheduled in each round for queue K . If the scheduler is not serving K when the packet x arrives, $L_k(t)$ th

scheduling interval of queue K will come $\frac{L_k(t_x)}{Q_k}$ rounds later. If K is under service when packet x arrives, the $L_k(t)$ th scheduling interval may come at most 1 round earlier i.e. after $\frac{L_k(t_x)}{Q_k} - 1$ rounds. Therefore, equation (4.5) guarantees that the tail packet α will be scheduled after packet β . DGCP condition 1 holds.

Step 2: Suppose the second pushout happens at t_1 . A packet of class C pushes out packet δ of class D . Since DGCP condition 1 holds before t_1 , theorem 1 and Corollary 1 also hold. The scheduling round of the current tail packet γ of class C is not earlier than $\frac{L_c(t_\gamma)}{Q_c} + R(t_\gamma) - 1$ using theorem 1. According to corollary 1, the scheduling round of packet δ will be no later than $\frac{L_d(t_\delta)}{Q_d}$. As a result, the tail packet of class C will be scheduled after packet δ . Condition 1 holds.

Step 3: Using the same logic, we can prove that if condition 1 holds after the i th cross pushout, condition 1 will still hold after $i+1$ th cross pushout. Given condition 1 holds at the beginning, it will always hold if equation (4.5) is used as the criteria to perform the cross pushout under WRR. Proof is complete.

Note that, equation (4.5) is in fact the equation (4.4) interpreted using the WRR scheduling round. Checking equation (4.5) may require recording and comparing the values of two real numbers since $\frac{L_k(t_x)}{Q_k}$ may not be integers. In implementation, using integer to record the departure round of a packet can save memory space. The sufficient condition provided by equation (4.5) to satisfy DGCP condition 1 can be released as:

$$\left\lceil \frac{L_a(t_\alpha)}{Q_a} + R(t_\alpha) \right\rceil > \left\lceil \frac{L_b(t_\beta)}{Q_b} + R(t_\beta) \right\rceil \quad (4.6)$$

With equation (4.5) or (4.6), we can implement DGCP under WRR scheduler. The architecture of the forwarding engine is shown in Fig. 4.7. We will discuss the implementation of the forwarding engine in detail in the following sections.

Figure 4.7 The architecture of a forwarding engine with DGCP under WRR scheduler.

4.6.1 Implementation of WRR Scheduler

The Weighted Round Robin Scheduler serves the logical queues in a round robin fashion. Assume there are N logical queues in the system, in each scheduling round, WRR serves queue 1 to queue N in turn. In each scheduling round, queue K can be served Q_k times if it has enough backlog. Otherwise, all the backlog of queue K will be cleared, the scheduler start serving the next logical queue. WRR is an $O(1)$ algorithm since each time it can schedule $Q_k \geq 1$ packets from a queue. To implement DGCP, WRR need to be modified in three aspects.

First, to use equation (4.5) or (4.6) to check DGCP condition 1, a mechanism is needed to records the service round $R(t)$. $R(t)$ is set to 0 at the beginning of each system busy time period and increments at the beginning of each round. At the end of a system busy period, after all backlogs are cleared, $R(t)$ will be reset to 0. A counter can be added to record the $R(t)$. Given the link will not always under congestion, $R(t)$ will not reach a very large number.

Second, when serving a packet, the scheduler behaves according to whether the packet is marked. If the packet is not marked, the scheduler sends the packet. Otherwise, the packet is dropped and the HOL packet from the queue that performed the pushout is checked. The process goes on until an unmarked HOL packet is found and sent out. The search process usually stops after searching 1 or 2 packets since: 1) The loss is a rare event even under congestion; 2) The number of queues in DiffServ schedulers are limited; 3) In certain period of time, pushout tends to happen in the same queue which has the best normalized loss ratio. Even under very rare case, the search process stops after more than 2 queues are checked, the scheduling still has an $O(1)$ complexity. The reason is from each packet's point of view, the scheduler only operates once, either send it or drop it.

Third, the scheduler will update the virtual backlog after packet departure. It is the queue whose scheduling interval is used that has its virtual backlog decremented.

The logic of serving the packet and updating the virtual backlog is shown in Fig. 4.3 in section 4.2. All the related operations such as dropping a packet, sending a packet and decrementing the virtual backlog can all be done with $O(1)$ complexity.

4.6.2 Implementation of Per Class Logical Queues

The logical queues are implemented using single link lists. Fig.4.8 shows the data structure of a logical queue.

Figure 4.8 Logical queue under DGCP.

The packets in each logical FIFO queue are linked in a single link list. The “head” pointer points to the head of line frame of a logical queue. Service always starts from the head. The “tail” pointer points to the tail frame of a logical queue. New packet always enters the link list at the tail. Different from the logical queue under a fair queueing system without DGCP, a third pointer called the “pushout” pointer is added to facilitate the pushout. It points to the unmarked frame that is queued at a position closest to the head. With this pointer, when performing cross pushout, the packet to be pushed out can be found easily without searching the list from the head at each time. The components in the single link list is called frame. It consists of three parts. The “data” part contains the packet or a pointer to the packet. The “mark” field of a frame records the ID of the queue that pushed out the packet. If the packet is not pushed out, the “mark” field is set to a default value that is different to any of the logical queue ID. The “round” field records the estimated departure round under DGCP when the packet arrives. If equation (4.7) is used,

the “round” could be a float number and is recorded as: $\frac{L_k(t_x)}{Q_k} + R(t_x)$. If equation (4.8)

is used, the “round” is an integer with value: $\left\lceil \frac{L_k(t_x)}{Q_k} + R(t_x) \right\rceil$. The overhead of the

frames can be reduced. The value of $R(t_x)$ is maintained by the WRR scheduler and can be obtained from the counter. The virtual backlogs $L_k(t_x)$ are maintained by both the WRR scheduler and the QMs.

4.6.3 Implementation of QM Schemes

Each logical queue in the system can have its own buffer allocation and QM. The only difference between QM under DGCP and the QM under a forwarding engine without cross pushout is the use of virtual backlog. Under DGCP, all QMs are based on virtual backlog. As discussed before, the WRR decrements the virtual backlog when a scheduling interval of the queue is passed. When QM admits a packet, it is QM’s responsibility to increment the virtual backlog and records the expected scheduling round of the packet using equation (4.5) or (4.6). Virtual backlog increments only when a packet is admitted by its QM. Cross pushout does not update the virtual backlogs. QM also triggers the cross pushout mechanism by sending a drop signal to the PLR dropper. All the above operations can be done with $O(1)$ complexity.

4.6.4 Implementation of PLR Dropper

The PLR dropper keeps records of the loss rates for service classes. It can be either $\text{PLR}(\infty)$ or $\text{PLR}(M)$. For $\text{PLR}(\infty)$, only two counters are needed to record the throughput

and loss respectively for each service class. To implement PLR (M), besides the two counters, a sliding window of size M is required to record the history of the packets loss for each queue. It can be easily implemented using array or linked list. One function of the PLR dropper is to find the queue with smallest normalized loss ratio. This operation can be done with $O(1)$ complexity if the loss records of all queues are listed in the increasing order of their normalized loss ratio.

4.6.5 Implementation of DGCP

With QM to record the expected departure round of each packet and PLR dropper to select the candidate queue to perform the cross pushout, the job left for DGCP is simple. It simply compares the departure rounds of two packets to check DGCP condition 1. If the condition is met, it marks the pushed out packet and enqueue the packet invoking the pushout. If the condition cannot be met, it returns a failure signal to PLR dropper. PLR dropper will select the queue with second smallest normalized ratio and inform DGCP to attempt a new pushout. The process ends until the cross pushout is successful or failed. All the operations such as marking or comparing the departure rounds can be done with $O(1)$ complexity. Considering the interactions between PLR dropper and DGCP, a cross-pushout will be checked at most N times. N is the number of the queues in the system. Since in DiffServ core network, each queue corresponds to a service class, the number of queues N is very small. The computational cost introduced by a single cross pushouts are limited.

We have given the complete implementation of a DGCP forwarding engine using WRR scheduler. Most of the operations are of $O(1)$ complexity. The only exception is the process of finding the right packet to perform the cross pushout has an $O(N)$ complexity. Since N , the number of queues in the DiffServ core router is limited and most importantly, the pushout is a rare event, the overall extra computation cost introduced by the new forwarding engine is neglectable. With DGCP, the new forwarding engine has the ability of providing hard QoS on delay and soft QoS on loss rates with only slightly extra computational cost.

4.7 DGCP with Variable Packet Sizes

In the previous sections, we have shown that in a system with fixed packet size, DGCP can adjust the loss rates while still guarantees the delay bounds. However, in real world, packet sizes are usually not fixed like in the ATM system. It is necessary to extend DGCP to handle packets with different sizes. For notational convenience, we will use DGCP+ to represent DGCP under variable packet sizes. In the following sections, we will first illustrate how DGCP+ works under Deficit Round Robin (DRR) scheduler and then verify the delay guarantee under DGCP+.

Here it is necessary to review the working principles of DRR scheduler. In DRR, to share the bandwidth, each queue i is assigned a quantum Q_i specifying how many bytes can be served at each scheduling round. Since in a packet-by-packet scheduling system only integer number of packets can be served in each round, the quantum Q_i may not be used up completely. Deficit is introduced to record the amount of service the scheduler

owes to the queue. In particular, if after the current round of service, the deficit has a value of D_i , in the next round, packets with $D_i + Q_i$ amount of bytes can be served. The deficit is set to zero at the beginning of a session busy period and is reset to zero when the session busy period ends. During a session busy period, the deficit is updated at the end of each scheduling round as following:

$$D_i = D_i + Q_i - W_i \quad (4.7)$$

W_i is the service queue i received in the scheduling round. By keeping the deficit, if a queue cannot use up its bandwidth in one round, it has the chance to make full use of its assigned bandwidth later. As a result, fairness is guaranteed.

Some important results of DRR are:

1. $0 \leq D_i < L_i \leq MTU$. The deficit size is always less than the maximum packet size of the queue and the system.
2. If queue i is continuously backlogged and it has been served for M scheduling rounds during time period (t_1, t_2) , the amount of service $W_i(t_1, t_2)$ it receives follows:

$$M \times Q_i - MTU < W_i(t_1, t_2) < M \times Q_i + MTU \quad (4.8)$$

3. Suppose the scheduler keeps record of its scheduling round during its system busy period. If when a packet of queue i arrives at time t , its queue length is $L_i(t)$ (including the size of the new arrived packet), and the scheduler round is $R(t)$, then its departure round R can be estimated as:

$$\left\lceil \frac{L_i(t) - MTU + 1}{Q_i} \right\rceil + R(t) \leq R \leq \left\lceil \frac{L_i(t) + MTU - 1}{Q_i} \right\rceil + R(t) \quad (4.9)$$

The left hand side shows the best case. Queue i has not started its service in round $R(i)$ and has a maximum deficit of $MTU-1$. The right hand side shows the worst case: queue i has finished its service in round $R(i)$ and no deficit is left currently. Furthermore, the new packet just missed one scheduling round because it has a MTU size and failing to schedule it leaves a deficit of $MTU-1$ bytes.

Now we can introduce cross pushout to DRR scheduler. Unlike DGCP where all packets have the same size, in DGCP+, in order to borrow enough bandwidth, one packet may need to push out multiple packets from another queue. The pushout conditions are redefined as following:

Upon arrival, each packet x of class K records its estimated worst-case departure round R_x as: $\left\lceil \frac{L_k^v(t_x) + MTU - 1}{Q_k} \right\rceil + R(t_x)$. Where $L_k^v(t_x)$ is the virtual backlog of class K and $R(t_x)$ is the scheduler's scheduling round at time t_x when packet x arrives. Q_k is the quantum of class K . A packet α of class A can pushout i packets $\beta_1, \beta_2 \dots \beta_i$ from class B, if and only if:

- 1) Suppose the original tail packet α_0 of queue A is estimated to be scheduled at round R_{α_0} . Packet β_i , the last packet of the pushed out packets, has an estimated scheduling round $R_{\beta_i} < R_{\alpha_0}$.
- 2) Packet $\beta_1, \beta_2 \dots \beta_i$ are not marked as already been pushed out by other packet(s).
- 3) $\sum_{j=1}^{i-1} L(\beta_j) < L(\alpha) \leq \sum_{j=1}^i L(\beta_j)$, where $L(.)$ is the length of the packet.

We shall see later that condition 1 and 2 ensure the bandwidth borrowed can be used timely. Condition 3 guarantees that necessary and sufficient amount of bandwidth is borrowed.

Although virtual backlogs in both DGCP+ and DGCP are designed to measure how many bytes will be served using the queue's own bandwidth, it is important to notice that they are different in the way they update. In DGCP, no update is needed after a cross pushout. However, under DGCP+ since the borrowed bandwidth in bytes may be larger than the size of the new packet, the virtual backlogs need to be updated as following:

$$\begin{aligned} L_a^v(t_\alpha) &= L_a^v(t_\alpha^-) + L(\alpha) - \sum_{j=1}^i L(\beta_j) \\ L_b^v(t_\alpha) &= L_b^v(t_\alpha^-) \end{aligned} \quad (4.10)$$

Equation (4.12) indicates that the queue borrowing bandwidth may have a smaller virtual backlog after a pushout since more bandwidth can be borrowed. The virtual backlog of the queue from which the bandwidth is borrowed keeps the same since all the amount of bandwidth borrowed can be used by the queue triggering the pushout.

After the pushout, when β_j 's scheduling turn comes, packet(s) equals to $L(\beta_j)$ amount of bytes can be scheduled from queue A using B's assigned bandwidth. After β_i 's scheduling turn, packet(s) equals to $\sum_{j=1}^i L(\beta_j)$ amount of bytes can be scheduled from queue A. If packets can be divided infinitesimally as in the fluid model, packets with exact $\sum_{j=1}^i L(\beta_j)$ bytes from queue A can be served. However, since in a packet-by-packet

scheduling system, packets are not dividable, the number of packets, k , can be served

using the borrowed bandwidth $\sum_{j=1}^i L(\beta_j)$ are :

$$\max(k) \sum_{j=1}^k L(\alpha_j) \leq \sum_{j=1}^i L(\beta_j) + D_a, \text{ if } L(\alpha_1) \leq \sum_{j=1}^i L(\beta_j) + D_a \quad (4.11)$$

$$0, \text{ if } L(\alpha_1) > \sum_{j=1}^i L(\beta_j) + D_a \quad (4.12)$$

Where D_a is the current deficit. Under DRR, the deficit is updated only at the end of each scheduling round according to equation (4.7). With DGCP+, since a queue can not only use its own bandwidth but also borrow bandwidth from other queues, it is necessary to extend D_i to record the deficit when the borrowed bandwidth is not used up. The deficit will also be updated every time after queue A is served using its borrowed bandwidth as following:

$$D_a = D_a + \sum_{j=1}^i L(\beta_j) - \sum_{j=1}^k L(\alpha_j) \quad (4.13)$$

Equation (4.11) and (4.12) guarantees that the amount of bytes served is less than the borrowed bandwidth plus the deficit. Equation (4.13) shows that the deficit is kept so that the borrowed bandwidth that is not used in this turn can be used later.

With the refined pushout behaviors, under DGCP+, we have:

Theorem 5: Under DGCP+, when packet α of queue K arrives at time t_α , suppose its virtual backlog is $L_k^v(t_\alpha)$ and the scheduling round is $R(t_\alpha)$, α records its estimated

departure round R_α as $\left\lceil \frac{L_k^v(t_\alpha) + MTU - 1}{Q_k} \right\rceil + R(t_\alpha)$. If there is no future arrival, the

departure round R of the tail packet α either follows:

$$R \leq R_\alpha = \left\lceil \frac{L_k^v(t) + MTU - 1}{Q_k} \right\rceil + R(t), \quad (4.14)$$

or, if (4.16) does not hold, R follows:

$$R \leq R_{\alpha 0}, \text{ where } \alpha 0 \text{ is packet queues just before } \alpha. \quad (4.15)$$

Specifically, if the maximum virtual backlog of queue K is L_k^v . We have:

$$R \leq \left\lceil \frac{L_k^v + MTU - 1}{Q_k} \right\rceil + R(t) \quad (4.16)$$

Proof: Let's denote the real queue length, the borrowed bandwidth in bytes, the virtual backlog of a queue K at time t to be $L_k^r(t)$, $L_k^b(t)$ and $L_k^v(t)$ respectively. The proof is similar to the proof of theorem 4 of DGCP.

Step 1: The virtual backlog $L_k^v(t)$ of class K is a step function of time t . Assume the first cross pushout happens at time $t0$. When $t < t0$, $L_k^v(t) \equiv L_k^r(t)$. As in DRR, the departure round of the packet follows equation (4.9). Theorem 5 holds.

Step 2: At time $t0^-$, a packet from class A pushes out a group of packet(s) from class B .

Three cases may happen:

1. $A=K$. The new packet α now becomes K 's tail packet and pushed out packets β_1 ,

$\beta_2 \dots \beta_i$ from class B . After the pushout, $L_k^r(t0) = L_k^r(t0^-) + L(\alpha)$,

$$L_k^b(t0) = L_k^b(t0^-) + \sum_{j=1}^i L(\beta_j), \quad L_k^v(t0) = L_k^r(t0) - L_k^b(t0) = L_k^v(t0^-) + L(\alpha) - \sum_{j=1}^i L(\beta_j).$$

Two cases may happen: 1) $\sum_{j=1}^i L(\beta_j)$ amount of bandwidth can be borrowed and used

before the new packet is served. As a result, all the borrowed bandwidth is guaranteed

to be used, and the packets going to be scheduled using the K 's own bandwidth is

clearly $L_k^v(t0)$. According to equation (4.9), equation (4.14) holds. 2) $\sum_{j=1}^i L(\beta_j)$

amount of bandwidth cannot be used up. This indicates that the backlog of queue K has been cleared before using up the borrowed bandwidth. Since DGCP+ condition 1 guarantees that packet β_i will be scheduled before $R_{\alpha0}$, both $\alpha0$ and α depart before $R_{\alpha0}$. Therefore, we have $R_\alpha \leq R_{\alpha0}$. Equation (4.15) holds. Since $R_{\alpha0}$ follows equation (4.14), R_α conforms to equation (4.16). Theorem 5 holds.

2. $B=K$. A group of packets $\beta_1, \beta_2 \dots \beta_i$ in K are pushed out. The round that the tail packet of K will be served (dropped if it is β_i) is not affected by the pushout. Virtual backlog of queue K is not updated. Theorem 5 holds.
3. K is neither A nor B . The service to all packets in class K is unchanged, theorem 5 holds.

Step 3: Suppose the next packet comes at time $t1^-$. Four cases may happen.

1. The packet enters class K directly. Two sub-cases may happen: 1) All the borrowed bandwidth will be used before the new packet α . The number of bytes that are going to be scheduled using K 's bandwidth now becomes $L_k^v(t1^-) + L(\alpha)$. Since $L_k^v(t1) = L_k^v(t1^-) + L(\alpha) \leq L_k^v$, according to equation (4.9), equation (4.14) and (4.16) hold. Theorem 5 holds. 2) Not all borrowed bandwidth can be used. Since DGCP+ condition 1 guarantees the scheduling time of the last borrowed bandwidth is earlier than $R_{\alpha0}$. If the backlog is cleared before $R_{\alpha0}$, we have: $R_\alpha \leq R_{\alpha0}$. Since $R_{\alpha0}$ conforms to equation (4.16), so will R_α . Theorem 5 holds.

2. The packet enters queue K by pushing out a group of packet $\beta_1, \beta_2 \dots \beta_i$ in queue B .

Queue B can be any queue other than queue K . Two cases may happen: 1) All the borrowed bandwidth can be borrowed and used before α is served. The packets that

are going to be served using K 's bandwidth is $L_k^v(t1) = L_k^v(t1^-) + L(\alpha) - \sum_{j=1}^i L(\beta_j)$.

According to equation (4.9), both equation (4.14) and (4.16) hold. Theorem 5 holds. 2)

Not all the bandwidth borrowed can be used. Since the scheduling time of the last packet using the borrowed bandwidth is guaranteed to be earlier than the calculated bound $R_{\alpha 0}$. If the backlog is cleared before $R_{\alpha 0}$, we have: $R_\alpha \leq R_{\alpha 0}$. Since $R_{\alpha 0}$ conforms to equation (4.16), so will R_α . Theorem 5 holds.

3. The new packet from queue A pushed out a group of packets $\beta_1, \beta_2 \dots \beta_i$ in queue K .

All other packets in queue K except $\beta_1, \beta_2 \dots \beta_i$ get served at their original scheduling intervals. The round that the tail packet of K will be served (dropped if it is β_i) is not affected by the pushout. Theorem 5 holds.

4. The new packet is neither a packet of class K nor does it enter its logical queue by pushing a packet of queue K . The service to all the packets in queue K is not affected.

Theorem 5 holds.

Suppose the future packets arrives at time $(t2, t3, t4, \dots)$, following the logic of step 3, we can prove theorem 5 holds till time $t2, t3, t4 \dots$. Therefore, theorem 5 will hold all the time. Proof complete.

Theorem 5 proves that if there is no future arrival, the tail packets will be scheduled before round R . R conforms to equation (4.16). Corollary 1 extends the bound of scheduling round when future arrivals are considered.

Corollary 1: A packet α of class K is guaranteed to be served no later than in round R , where R is calculated as following:

$$R_{\alpha} \leq \left\lceil \frac{L_k^v + MTU - 1}{Q_k} \right\rceil + R(t) \quad (4.16)$$

Proof: When packet α of class K arrives, it is the tail packet of K . If no future arrivals, from theorem 5, equation (4.16) holds.

Considering the future arrivals, the departure rounds of the already backlogged packets of class K may change under two cases:

- 1) Packet $\gamma_1, \gamma_2, \gamma_i$ queued before α of class K are pushed out. The departure round of packet α is not affected. Corollary 1 holds.
- 2) New packet of K pushes out a group of packet $\beta_1, \beta_2 \dots \beta_i$ that will be scheduled before α . Then, including α , the packets in K which will be scheduled after β_i will get service earlier. Packet α will depart in a round earlier or equals to its estimated worst-case departure round R_{α} . Corollary 1 holds.

Combine case 1 and 2, proof is complete.

Corollary 1 shows that a packet's worst case departure round can be estimated using equation (4.16) at the time it arrives. Under DGCP+, the future arrivals will not delay the service to the already backlogged packets. In fact, a packet may be served earlier due to the bandwidth "borrowed" by future pushouts.

Corollary 1 in DGCP + is similar to corollary 1 in DGCP in that they all guarantee the scheduling round (scheduling interval in DGCP) of a packet. In DGCP, since all the packets have the same length, every time a packet is pushed out, another packet will use the borrowed bandwidth completely. The number of packets transferred in a scheduling

round does not change because of the cross pushouts. Therefore, guaranteeing the scheduling round (scheduling interval) under DGCP can actually guarantee the packet delay since each scheduling round (scheduling interval) takes the same time. Under DGCP+, the amount of packets can be sent using the borrowed bandwidth follows equation (4.11) and (4.12). Usually, the bytes sent and the bandwidth borrowed do not equal. As a result, the number of bytes sent in each scheduling round (also called frame length in round robin scheduler) varies due to the cross pushout. Guaranteeing the scheduling round, as the result of corollary 1, may not bound the delay since the time each round takes may vary. We cannot declare delay guarantee without comparing the frame lengths in DRR with and without DGCP+.

Let us review the frame length under DRR without DGCP+. Suppose M complete rounds $R_{k+1}, R_{k+2} \dots R_{k+m}$ has passed. Under the worst case, each queue always has enough packets to be sent and keeps backlogged during all the M rounds. The total amount of packets sent in the M rounds follows:

$$\sum_{i=1}^n W_i = \sum_{i=1}^n (M \times Q_i + D_i^k - D_i^{k+m}) \leq M \sum_{i=1}^n Q_i + n \times (MTU - 1) \quad (4.17)$$

Where n is the number of queues in the system. D_i^k is the deficit of queue I at the beginning of round $k+1$, D_i^{k+m} is the deficit of queue I after round $k+m$. The second inequality follows the fact that the deficit is always between 0 and $MTU-1$.

The frame length under DGCP+ is more complicated since bandwidth can be borrowed among the queues. Let's denote the sum of bandwidth borrowed by queue J from queue I in round k as $W_{i,j}^k$, the number of bytes served in queue I in round k as W_i^k . Then we have, in any round k , the service to packets in queue I follows:

$$W_i^k = Q_i + \sum_{j=1}^n W_{j,i}^k - \sum_{j=1}^n W_{i,j}^k + D_i^{k-1} - D_i^k \quad (4.18)$$

Equation (4.18) shows the service to packets in a queue roughly equals to its assigned quantum plus the bandwidth it borrowed from other queues minus the bandwidth it lent to other queues. The discrepancy is kept in the deficit.

With equation (4.18), we have the frame length of the round k follows:

$$\begin{aligned} \sum_{i=1}^n W_i^k &= \sum_{i=1}^n Q_i + \sum_{i=1}^n \sum_{j=1}^n W_{j,i}^k - \sum_{i=1}^n \sum_{j=1}^n W_{i,j}^k + \sum_{i=1}^n (D_i^{k-1} - D_i^k) \\ \sum_{i=1}^n W_i^k &= \sum_{i=1}^n (Q_i + D_i^{k-1} - D_i^k) \end{aligned} \quad (4.19)$$

Then we have the length of M rounds under DGCP+ follows:

$$\sum_{i=1}^n W_i = \sum_{K=k+1}^{k+m} \sum_{i=1}^n W_i^K = \sum_{i=1}^n (M \times Q_i + D_i^k - D_i^{k+m}) \leq M \sum_{i=1}^n Q_i + n \times (MTU - 1) \quad (4.20)$$

Equation (4.20) gives the upper bound of the length of M frames under DGCP+. It is identical to the bound given in (4.17). Therefore, by guaranteeing the scheduling round, the same worst-case delay bound can be guaranteed with or without DGCP+ under DRR. With Corollary 1 and equation (4.17) and (4.20), we can claim the following result:

Corollary 2: If a worst-case delay bound is guaranteed under DRR by limiting the queue length. Controlling the virtual backlog under the same limit in DGCP+ will guarantee the same worst-case delay bound.

Corollary 2 shows us the same delay bound can be guaranteed under DGCP+ if the pushout conditions are met. Now with some minor modifications, DGCP can be upgraded to DGCP+ and the ability to adjust loss rates while guaranteeing the delay bounds is kept.

Similar question on the total loss may be raised under DGCP+. Recall theorem 3 in DGCP, the same total loss can be guaranteed regardless of the cross pushout since the

virtual backlogs in system 3 always equal to the real queue lengths in system 1. However, theorem 3 does not hold in DGCP+. Due to the variable packet sizes, under DGCP+ one packet may push out multiple packets and the virtual backlog changes after the cross pushout. The total loss will not remain the same because: 1) according to DGCP+ condition 3 the amount of bytes pushed out can be bigger than the size of the packet admitted. 2) Virtual backlog becomes shorter if more bytes than the size of admitted packet are pushed out. System 1 and system 3 are not synchronized. It is analytically difficult to compare the total loss with or without cross pushout under DGCP+ because of the dynamics of traffic and real queue lengths/virtual backlogs. However, the total loss of the two systems can be expected to be similar. Although for each individual cross pushout it seems more loss is introduced due to fact that the amount of bytes pushed out can be more than the amount of bytes admitted, the virtual backlog after the pushout becomes shorter and leads to smaller chance of future pushouts. We will study the total loss under DGCP+ using simulation in Chapter 5.

Chapter 5

Simulation Result

To test the performance of DGCP and DGCP+, we simulate DGCP and DGCP+ using OPNET simulator. The network simulated has a single bottle-neck topology.

Figure 5.1 Simulated network topology.

The network topology is shown in Fig.5.1. There are n source-destination pairs. Traffic generated by source node S_i has a destination D_i . The only bottle-neck link in the network stays between router0 and router1. The proposed forwarding engine is implemented in

router0 to provide differentiated services to service classes when they compete the resources on the congested bottle-neck link.

We compare the performance under three parallel systems in router 0. All three systems have the same input traffic and parameter settings for queue management and scheduling algorithms.

System 1 (architecture shown in Fig.3.2.): The original fair queueing based packet forward engine without any loss control and cross pushout.

System 2 (architecture shown in Fig.3.3.): PLR dropper is applied directly to the fair queueing system. No bandwidth “borrowing” after pushout.

System 3 (architecture shown in Fig.4.5.): DGCP or DGCP+ is added to the fair queueing system.

5.1 Delay Guarantee under DGCP

Experiment 1 is aimed to verify DGCP property 1: the delay guarantee under DGCP. We have 5 sources all sending exponential traffic with an average rate of 20% of the bottleneck link speed. The scheduler is WRR and bandwidth assigned to each queue is 20% of the link speed. The loss rate differentiation parameters in system 2 and 3 are all set to the same to achieve similar loss rates for all service classes. RED is used as the QM scheme for each queue with the parameters set as following:

Table 5.1QM Parameters in Experiment 1

Queue	Min threshold	Max threshold	Max Drop probability	RED weight
Queue 1	16	20	0.1	1
Queue2-5	40	50	0.1	1

The results of experiment 1 are shown in Fig. 5.2

Figure 5.2 Loss rates under three systems.

Fig. 5.2 shows the loss rates with 90% confidence interval of all 5 queues during the simulation. Without loss control, under system 1, queue 1 experiences higher loss rate due to smaller buffer space. The loss rates of queue 2-4 are similar.

The loss rate differentiation parameters are all set to 1 under both system 2 and 3 to achieve similar loss rates targets of all five queues. As expected, with loss control, the loss rates of all queues become similar. However, loss rates in system 2 are significantly higher than in system 3. The main reason is that under system 2 the pushout does not associated with bandwidth borrowing. Although a packet can enter the queue by pushing out another packet, the delay to this packet is not necessarily improved. A queue's real queue length increases after the cross pushout. As a result, future arrivals of the same queue have a higher probability to see a large queue length and therefore trigger more

pushouts. On the other hand, in DGCP a cross pushout always leads to bandwidth borrowing. The virtual backlogs do not change after each cross pushout. DGCP Theorem 2 indicates that the total loss is fixed no matter DGCP is used or not. DGCP will not cause more loss. This explains that although System 2 and DGCP both have the ability to control loss rates to meet the relative service differentiation targets, the absolute loss rates in DGCP are much lower.

Figure 5.3 CDFs of packets delay under three systems.

Fig. 5.3 and 5.4 show the CDF of packet delays in queue 1 under 3 systems. The delays under system 1 are bounded by the buffer size and the allocated bandwidth. More than 1% of the packets under system 2 have longer delays than the delay bound under system 1. The delay violations are introduced by the cross pushout. Since under system 2, the cross pushout does not associated with bandwidth borrowing, the buffer size limit is

broken and leads to longer delay. Considering the loss rate of queue 1 under system 2 (shown in Fig.5.2) is only 0.9%, the rate of delay bound violation is significant. Under system 3, as expected, all the packets follow the same delay bound as in system 1. Therefore, as proven in theorem 1, DGCP can guarantee the packets delay.

Figure 5.4 Tail part of CDFs of packets delay under three systems

5.2 Total Loss under DGCP

The objective of experiment 2 is to verify DGCP property 2: The total loss under DGCP is fixed if the input traffic is not affected by the loss.

The system contains two sources both generate exponential traffic with an average rate of 50% of the link speed. Buffer size and the bandwidth allocation are the same for both queues. In system 1, there is no loss control; we run the simulation once to collect the

total loss of both queues. In system 2 and 3, we fix the loss rate differentiation parameter σ_1 of queue 1 and change the loss rate differentiation parameter σ_2 of queue 2 to see the how the total loss will change accordingly. The results of experiment 2 are shown in Fig. 5.5, 5.6 and 5.7.

Figure 5.5 Total loss under system 2 and system 3.

Fig. 5.5 shows the total loss of both queues under different loss rate differentiation parameters. The total loss is collected with about 200K packets sent in total. As expected, the total loss under DGCP is exactly the same as the total loss under system 1 and does not change under different loss rate differentiation parameters. Without the DGCP, system 2 has a bigger total loss that varies each time the rate differentiation parameters change. The tail part of the total loss rate curve under system 2 seems flat. However, it is not because the total loss is less correlated with the loss differentiation targets but

because the target loss rate differentiation region becomes unreachable as we shall see later.

Although the loss rate differentiation parameters σ_i can be set to any value in the proportional differentiated service model, the target proportional loss rate differentiation may not be reached. Pushout fails when a packet from class j intends to push out a packet from class i but there is no unmarked packet in class i . The achievable loss rate differentiation region is limited and different for each scheme.

Figure 5.6 Achievable loss rate differentiation region under system 2.

Fig. 5.6 shows the actual loss rate ratio $L1/L2$ against the target ratio σ_1/σ_2 in system 2. Under the current system settings, the maximum loss ratio that can be achieved without DGCP is less than 4.

Figure 5.7 Achievable loss rate differentiation region under DGCP.

Fig. 5.7 shows the achievable loss rate differentiation region under DGCP. We can see that the maximum actual loss ratio is above 25. Furthermore the actual ratio follows the target ratio nearly linearly until it reaches 10. Therefore, with DGCP, much larger quality spacing can be achieved. The major reason the controllable region is smaller without DGCP is that a pushout is not necessarily associated with a bandwidth borrowing and therefore may not result in a gain.

5.3 Delay Guarantee under DGCP+

The goal of experiment 3 is to verify theorem 5 and its corollaries: the delay guarantee under DGCP+. In the experiment, DRR scheduler is used to provide fair queuing under variable packet sizes. Both the loss and throughput are counted in bytes to take different

packet sizes into consideration. The buffer size, packet size distribution, packet inter-arrival time distribution, bandwidth allocation and loss rate differentiation parameter are set to be different for each queue in order to verify the delay guarantee under general conditions. The parameters are set as following:

Table 5.2 Simulation Parameters in Experiment 3.

Class	Packet size (bytes)	Inter-arrival time (second)	Buffer Size (bytes)	Quantum (bytes)	Loss rate differentiation parameter
1	Poisson (500)	Exp(0.025)	25,000	2,000	2.0
2	Poisson (250)	Exp(0.0125)	20,000	1,900	1.0
3	Poisson (500)	Exp(0.025)	15,000	2,100	1.0
4	Constant(500)	Exp(0.025)	25,000	2,000	1.0
5	Constant(500)	Exp(0.025)	25,000	2,000	1.0

Figure 5.8 Loss rates under two systems.

Fig 5.8 shows the loss rates with 90% confidence interval under system 1 and 3 respectively. Bucket method is used to collect the data in one simulation run. During each bucket, the total packets served are about 50M bytes. Altogether, 100 buckets of data are collected. In system 1, without the loss control, the loss rates of different queues are different. In system 3, with DGCP+, the loss rates are controlled according to the loss rate differentiation parameters.

Figure 5.9 The tail part of packets delay CDFs.

Fig. 5.9 shows the tail parts of CDFs of packet delays under system 1 and system 3. Without loss of generality, we only show the packet delays of queue 2. We can see clearly that the maximum delay experienced under DGCP+ is no bigger than the maximum delay under system 1. Therefore, the delay bounds are guaranteed.

5.4 Total Loss under DGCP+

Experiment 4 is set up to test the difference in total loss between system 1 and system 3. The parameter settings are exactly the same as experiment 3 as shown in Table. 4. Bucket method is used in the simulation. For each bucket, the total loss of five queues is collected in a period during which about 50M bytes of packets are served. The simulation contains 100 buckets. Fig.5.10 shows the results. The x-axis and y-axis show the loss under system 1 and system 3 respectively. We can see in the figure, all the sample values are concentrated at the $x=y$ curve. The numbers of total lost packets under both systems are very close to each other. Since the applications can usually tolerate certain loss rate, such a close total loss rate under DGCP+ should be considered acceptable.

Figure 5.10 Total loss under system 1 and system 3.

Chapter 6

Conclusions and Recommendations for Future Research

6.1 Conclusions

In this thesis we propose a Delay Guaranteed Cross Pushout scheme. To our best knowledge, it is the first scheme to provide loss rate control among queues under fair queueing schedulers. Two important properties of DGCP are verified by both mathematical proof and simulations:

- 1) DGCP guarantees delay bounds.
- 2) DGCP keeps the total loss fixed.

DGCP+ is proposed to work under DRR scheduler to handle packets with variable sizes. The delay guarantee is still provided under DGCP+. Moreover, through simulation, we show the total loss under DGCP+ is maintained to be very close to the total loss when no loss control is applied.

DGCP and DGCP+, can work with fair queueing schedulers smoothly to form a new packet forwarding engine with little modification. By using the word “smoothly” we mean that under the new forwarding engine, the hard QoS on packets delay provided by the original fair queueing schedulers and queue management schemes are kept unchanged.

Meanwhile the new forwarding engine has the ability to provide soft QoS on loss rates. It is the first forwarding engine that based on fair queueing scheduler that can provides both hard QoS on packet delay and soft QoS on packets loss at the same time.

6.2 Recommendations for Future Research

This thesis provides a mechanism to achieve the proportional loss rates differentiation target among different queues under fair queueing schedulers. However, DGCP is not limited to be used with PLR dropper to provide proportional loss rate differentiation. In fact, it could work with any loss controller. The next stage of research will be focused on applying DGCP to meet more specific needs of real applications. For example:

- 1) Within one queue, different loss targets maybe set for packets with different loss preferences. For instance, in video transmission service, the loss rates targets for packets belong to different frame types (I, B, P) are different. Mechanisms that can adjust loss rates not only inter but also intra classes will be studied next.
- 2) The loss rates differentiation targets can be set according to the applications need. For example, in video transmission, a performance metrics may be set up according to the relation between the play back quality and the loss rates of different type of packets. A loss rate controller aims to achieve the optimal loss rates targets may be developed to work with DGCP/DGCP+ to provide the best playback quality.

References

- [1] S.Blake, D.Black, M.Carlson, E.Davies, Z.Zhang, W.Weiss, "An architecture for differentiated services." IETF RFC 2475. Dec 1998.
- [2] V. Jacobson, K. Nichols, K. Poduri, "An expedited forwarding PHB." IETF RFC 2598. Jun. 1999.
- [3] J. Heinanen, F. Baker, W. Weiss, J.Wroclawski, "Assured forwarding PHB group." IETF RFC 2597. Jun. 1999.
- [4] A.K. Parekh, R.G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: the single node case," IEEE/ACM Transactions on Networking, Pages: 344 - 357. Jun. 1993.
- [5] A.K Parekh, R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case" IEEE/ACM Transactions on Networking, Volume: 2 , Issue: 2 , Pages:137 – 150. Apr. 1994.
- [6] M. Shreedhar, G. Varghese, "Efficient fair queuing using deficit round-robin." IEEE/ACM Transactions on Networking, Volume: 4, Issue: 3, Pages: 375 - 385. Jun. 1996.
- [7] M. Katevenis, S. Sidiropoulos, C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip" IEEE Journal on Selected Areas in Communications, Volume: 9, Issue: 8, Pages: 1265 – 1279. Oct. 1991.
- [8] D. Stiliadis, A. Varma, "Efficient fair queueing algorithms for packet-switched networks" IEEE/ACM Transactions on Networking, Volume: 6, Issue: 2, Pages:175 – 185. Apr. 1998.
- [9] D. Stiliadis, A. Varma, "Rate-proportional servers: a design methodology for fair queueing algorithms" IEEE/ACM Transactions on Networking, Volume: 6, Issue: 2, Pages:164 – 174. Apr. 1998.
- [10] S. Suri, G. Varghese, G. Chandranmenon, "Leap forward virtual clock: a new fair queueing scheme with guaranteed delays and throughput fairness" INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Volume: 2, Pages: 557 - 565. Apr. 1997.
- [11] J.C.R. Bennett, H. Zhang, "WF²Q: worst-case fair weighted fair queueing" INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies.

Networking the Next Generation. Proceedings IEEE, Volume: 1, Pages: 120 - 128. March 1996.

[12] J.C.R. Bennett, D.C. Stephens, H. Zhang, "High speed, scalable, and accurate implementation of packet fair queueing algorithms in ATM networks" International Conference on Network Protocols, Pages: 7 – 14 Oct. 1997.

[13] J.C.R. Bennett, H. Zhang, "Hierarchical packet fair queueing algorithms" IEEE/ACM Transactions on Networking, Volume: 5, Issue: 5, Pages: 675 – 689. Oct. 1997.

[14] I. Stoica, S. Shenker, H. Zhang, "Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high-speed networks" IEEE/ACM Transactions on Networking, Volume: 11, Issue: 1, Pages:33 – 46. Feb. 2003.

[15] I. Stoica, H. Zhang, S. Shenker, "Self-verifying CSFQ" INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Volume: 1, Pages: 21 - 30. Jun. 2002.

[16] Z. Cao, Z. Wang, E. Zegura "Rainbow fair queueing: fair bandwidth sharing without per-flow state"INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Volume: 2, Pages: 922 – 931. Mar. 2000.

[17] S. Ramabhadran, J. Pasquale, "Stratified round robin: a low complexity packet scheduler with bandwidth fairness and bounded delay" SIGCOMM '03, Pages: 239-250. Aug. 2003.

[18] H. Shimonishi, L. Maki, T. Murase, M. Murata, "Dynamic fair bandwidth allocation for DiffServ classes.", IEEE International Conference on Communications, Volume: 4,. Pages: 2348 – 2352. Apr. 2002.

[19] S. Floyd, V. Jacobson, "Link-sharing and resource management models for packet networks" IEEE/ACM Transactions on Networking, Volume: 3, Issue: 4, Pages: 365 – 386. Aug. 1995

[20] D. Ferrari, D.C. Verma, "A scheme for real-time channel establishment in wide-area networks", IEEE Journal on Selected Areas in Communications, Volume: 8, Issue: 3, Pages:368 – 379. Apr. 1990.

[21] D.E. Wrege, J. Liebeherr, "A near-optimal packet scheduler for QoS networks" Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Volume: 2, Pages:576 – 583. Apr. 1997

- [22] J. Liebeherr, D.E. Wrege, D. Ferrari, "Exact admission control for networks with a bounded delay service", IEEE/ACM Transactions on Networking, Volume: 4, Issue: 6, Pages: 885 – 901. Dec. 1996.
- [23] L. Georgiadis, R. Guerin, V. Peris, K.N. Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping", IEEE/ACM Transactions on Networking, Volume: 4, Issue: 4, Pages: 482 – 501. Aug. 1996.
- [24] F.M. Chiussi, V. Sivaraman, "Achieving high utilization in guaranteed services networks using early-deadline-first scheduling", Sixth International Workshop on Quality of Service, Pages: 209 – 217. May 1998.
- [25] V. Sivaraman, F.M Chiussi, M. Gerla, "Traffic shaping for end-to-end delay guarantees with EDF scheduling", Eighth International Workshop on Quality of Service, Pages: 10 – 18. Jun. 2000.
- [26] V. Sivaraman, F.M Chiussi, M. Gerla, "End-to-end statistical delay service under GPS and EDF scheduling: a comparison study" Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Volume: 2, Pages: 1113 – 1122. Apr. 2001.
- [27] K. Pyun, J. Song, H.K. Lee "The service curve service discipline with the service curve service discipline for the rate-controlled EDF service discipline in variable-sized packet networks" IEEE International Conference on Communications, Volume: 2, Pages: 1135-1141. Apr. 2002.
- [28] Y. Chen, C. Qiao, M. Hamdi, D.H.K. Tsang, "Proportional differentiation: a scalable QoS approach." Communications Magazine, IEEE, Volume: 41, Issue: 6, Pages: 52 – 58. Jun. 2003
- [29] C. Dovrolis, D. Stiliadis, P. Ramanathan, "Proportional differentiated services: delay differentiation and packet scheduling." IEEE/ACM Transactions on Networking, Volume: 10, Issue: 1, Pages: 12 - 26. Feb. 2002.
- [30] C.Li, S. Tsao, M. Chen, Y. Sun, Y. Huang, "Proportional delay differentiation service based on weighted fair queuing", Ninth International Conference on Computer Communications and Networks, Pages: 418 – 423. Oct. 2000.
- [31] H. Saito, C. Lukovszki, I. Moldovan, "Local optimal proportional differentiation scheduler for relative differentiated services", Ninth International Conference on Computer Communications and Networks, Pages: 544 – 550. Oct. 2000.
- [32] Y. Lai, W. Li, "A novel scheduler for proportional delay differentiation by considering packet transmission time", Communications Letters, IEEE, Volume: 7, Issue: 4, Pages: 189 – 191. Apr. 2003.

- [33] H. Ngin, C. Tham, "Achieving proportional delay differentiation efficiently", 10th IEEE International Conference on Networks, Pages: 169 – 174. Aug. 2002.
- [34] M.K.H. Leung, J.C.S. Lui, D.K.Y Yau, "Adaptive proportional delay differentiated services: characterization and performance evaluation", IEEE/ACM Transactions on Networking, Volume: 9, Issue: 6, Pages: 801 – 817. Dec. 2001.
- [35] C. Dovrolis, P. Ramanathann, "Proportional differentiated services, part II: loss Rate differentiation and packet dropping", Eighth International Workshop on Quality of Service. Pages: 53 –61. 5-7 Jun. 2000
- [36] A.Y.-M. Lin, J.A. silvester, "Priority queueing strategies and buffer allocation protocols for traffic control at a ATM integrated broadcast system." IEEE journal on selected areas in communication, Volume 9, Pages: 1524-1536. Dec.1991.
- [37] N.L.S. Fonseca, J.A. Silvester, "A comparison of push-out policies in an ATM multiplexer", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Volume: 1, Pages: 338 - 341. May 1993.
- [38] H. Kroner, G. Hebuterne, P. Boyer, A. Gravy, "Priority management in ATM switching nodes" IEEE journal on selected areas in communication, Volume 9, Pages: 418-427, Apr.1991.
- [39] S. Floyd, V. Jacobson, "Random early detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking, Volume: 1, Issue: 4, Pages: 397 – 413. Aug. 1993.
- [40] Technical Specification from Cisco, "Distributed weighted random early detection." URL: <http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/c111/wred.pdf>.
- [41] D.D. Clark, W. Fang, "Explicit allocation of best effort packet delivery service." IEEE/ACM Transactions on Networking, Volume 6, Pages: 362-373, Aug. 1998.
- [42] S. Athuraliya, S.H. Low, V.H. Li, Q.Yin, " REM: active queue management" Network, IEEE, Volume: 15, Issue: 3, Pages: 48 – 53. May 2001.
- [43] T.J. Ott, T.V. Lakshman, L.H. Wong, "SRED: stabilized RED" INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Volume: 3, Pages: 1346 – 1355. 21-25 March 1999.
- [44] R. Pan, B. Prabhakar, K. Psounis, "CHOKe - a stateless active queue management scheme for approximating fair bandwidth allocation" Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Volume: 2, Pages: 942 – 951. Mar. 2000.

[45] W. Feng; K.G. Shin, D.D. Kandlur, D. Saha, "The BLUE active queue management algorithms", IEEE/ACM Transactions on Networking, Volume: 10, Issue: 4, Pages: 513 – 528. Aug. 2002.