MINIMIZING QUEUEING DELAYS

IN

COMPUTER NETWORKS

NGIN HOON TONG

(B.Eng(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2002

Acknowledgements

I wish to express my gratitude to my supervisor, Dr. Tham Chen Khong, for his invaluable guidance, support, encouragement, understanding and time, throughout my studies. He was the one who convinced me to get a doctoral degree and this thesis is a result of his successful persuasion and mentoring.

My time at the Computer Communication Networks Laboratory had been enjoyable because of friends and colleagues. This was cut short by departure to complete my National Service liabilities and subsequently my work commitments. Special thanks must be given to Mr. Gan Yung Sze and Dr. Jiang Yuming for their simulating discussions and constructive comments. In particular, Yung Sze has helped to review many of my earlier paper submissions.

I would like to express my earnest gratitude to my family for their love, support, and encouragement, without which any of my achievements would not have been possible. Thanks to my father and mother, whose love and countless sacrifices to raise and give me the best possible education gave me the strength to overcome any difficulties necessary to complete this degree. Thanks to my brother and sister for their support and encouragement. Last but not least, thanks to my dear wife Hsin Ning for her love

and understanding throughout this journey. She was always behind me and gave her unconditional support even if that meant to sacrifice the time we spent together. I delicate this thesis to my family.

Contents

Summary

The current Internet provides a best-effort packet service using the Internet Protocol (IP). It offers no guarantees on actual packet deliveries and users need not make reservations before transmitting packets through it. This architecture has been tremendously successful in supporting data applications as demonstrated by the remarkable growth of the Internet usage over the last decade. However, as the Internet evolves to become a global communication infrastructure, two key weakness have become increasingly obvious. Firstly, it is unable to provide service differentiation so that the network can utilize resources more efficiently to support the many new real-time applications that have started to proliferate over the Internet. Secondly, there is a lack of flow isolation within aggregated traffic which allows congestion unresponsive flows, such as User Datagram Protocol (UDP) flows, to squeeze out the congestion responsive ones, such as Transmission Control Protocol (TCP) flows.

This thesis addresses the key deficiencies of the best-effort paradigm through the proposal of an original service differentiation framework, called the Delay-Rate Differentiated Services (DRDS). The DRDS framework consists of two portions that provide delay based service differentiation and flow isolation within best-effort traffic respec-

tively.

The first portion addresses the issue of inefficient resource utilization by providing delay based service differentiation among classes of traffic aggregates. It is based on the Delay-Rate Differentiation (DRD) model, which refines on the Proportional Delay Differentiation (PDD) model, proposed by Dovrolis. The DRD model is a combination of the PDD model with another proposed model, called the Generalized Minimum Queueing Delay (GMQD) model. The PDD is a model that provides delay-based proportional differentiation among backlogged service classes traversing a single link. The GMQD is a model that minimizes the total queueing delay of all backlogged service classes traversing a single link. Depending on traffic load conditions, DRD is able to switch between PDD and GMQD, thus exploiting the advantages of both models. Two classes of packet scheduling algorithms emulating GMQD and DRD are also proposed and analyzed. A novel approximation technique that reduces the computational complexity of one class of algorithms proposed for GMQD and DRD is subsequently proposed. This technique reduces the computational complexity of the scheduling algorithms without comprising on their performance.

The second portion complements the first portion by addressing the issue of congestion responsive flows versus congestion unresponsive flows within each class of traffic aggregates. A novel control-theoretical approach, which enhances the flow isolation performance of existing fair queueing algorithms that do not maintain per flow state information is proposed.

List of Figures

List of Tables

Abbreviations

ABE	Alternative Best-Effort
ADDs	Average Drop Distances
AF	Assured Forwarding
ATM	Asynchronous Transfer Mode
BB	Bandwidth Broker
BPR	Backlog Proportional Rate
BEDS	Best-Effort Differentiated Services
CSFQ	Core-Stateless Fair Queueing
DDP	Delay Differentiation Parameter
DiffServ	Differentiated Services
DP	Dynamic Programming
DPS	Dynamic Packet State
DRD	Delay-Rate Differentiation
DRR	Deficit Round Robin
DSCP	Differentiated Services Code Point
DS field	Differentiated Services Field

DWFQ	Dynamic Weighted Fair Queueing
EF	Expedited Forwarding
Ex-VC	Extended Virtual Clock
FIFO	First-In-First-Out
GMQD	Generalized Minimum Queueing Delay
GPS	Generalized Processor Sharing
IETF	Internet Engineering Task Force
IntServ	Integrated Services
JoBS	Joint Buffer Management and Scheduling
LIRA	Location Independent Resource Allocation
LQ	Linear Quadratic
MAN	Metropolitan Area Network
MPLS	Multi-Protocol Label Switching
PDD	Proportional Delay Differentiation
PDM	Proportional Differentiation Model
PDRD	Packetized Delay Rate Differentiation
PFQ	Packet Fair Queueing
PGMQD	Packetized Generalized Minimum Queueing Delay
PHB	Per-Hop Behavior
PLD	Proportional Loss Differentiation
PLR	Proportional Loss Rate
PQCM	Proportional Queue Control Mechanism

QD-PDRD	Queueing Delay based Packetized Delay Rate Differentiation
QL-PDRD	Queue Length based Packetized Delay Rate Differentiation
QD-PGMQD	Queueing Delay based Packetized Generalized Minimum Queueing Delay
QL-PGMQD	Queue Length based Packetized Generalized Minimum Queueing Delay
QoS	Quality of Service
RED	Random Early Drop
RFQ	Rainbow Fair Queueing
RSVP	Resource Reservation Protocol
SCORE	Stateless Core
SDP	Scheduler Differentiation Parameter
SFQ	Start-Time Fair Queueing
SQD-PDRD	Scaled Queueing Delay based Packetized Delay Rate Differentiation
SQD-PGMQD	Scaled Queueing Delay based Packetized Generalized Minimum Queueing Delay
STP	Scaled Time Priority
TCP	Transmission Control Protocol
TDP	Time Dependent Priorities
TOS	Type-Of-Service
TUF	Tag-based Unified Fairness

UDP	User Datagram Protocol
VTRS	Virtual Time Reference System
WAN	Wide Area Network
WTP	Waiting Time Priority

Chapter 1

Introduction

1.1 The Best-Effort Service Paradigm of the Internet

The current Internet provides a best-effort packet service using the Internet Protocol (IP) [1]. It offers no guarantees on actual packet deliveries and users need not make reservations before transmitting packets through it. This architecture has been tremendously successful in supporting data applications as demonstrated by the remarkable growth of the Internet usage over the last decade. However, as the Internet evolves to become a global communication infrastructure, two key weakness have become increasingly obvious.

1.1.1 Inefficient Network Resource Utilization

The first key weakness is the inability to provide service differentiation so that the network can utilize resources more efficiently to support the many new real-time applications that have started to proliferate over the Internet. These applications, like Internet telephony and distributed interactive online-games, require different service levels

due to specific Quality-of-Service (QoS) requirements. Currently, applications with low QoS requirements, like e-mail, and applications with demanding QoS requirements, like Internet telephony, get the same QoS treatment in the router queues.

Naturally, network operators can provide an adequate performance to any demanding applications if they over-provisioned their routers and links. However, from an economic point of view, this means that they are not efficiently utilizing their network resources. This can be especially significant if the forwarding resources are expensive, like satellite connections.

On the other hand, when the network operators do not have sufficient forwarding resources at their routers and links, then only the less demanding applications can have adequate performance. If users of demanding applications are willing to pay a substantial premium to network operators who can deliver good performance to their demanding applications, then it will make good economic-sense for the network operators to allocate their network resources to these premium-paying users. However, this is not possible with the existing best-effort paradigm of the Internet.

1.1.2 Lack of Flow Isolation Between Congestion Responsive Flows and Congestion Unresponsive Flows

The second key weakness is the lack of flow isolation between congestion responsive flows and congestion unresponsive flows. The current Internet relies heavily on end-hosts implementing end-to-end congestion control mechanisms, in which end-hosts reduce their transmission rate under network congestion, to prevent network "meltdown". The most

widely utilized form of end-to-end congestion control mechanism is the Transmission Control Protocol (TCP) [2]. However, not all traffic flows include congestion avoidance mechanisms, either deliberately or due to incorrect implementation of the congestion avoidance algorithm. Furthermore, there are other transport layer protocols, like User Datagram Protocol (UDP) [3] that do not back off under congestion. As a result, these congestion unresponsive flows tend to use up bandwidth more aggressively, squeezing out the congestion responsive flows.

This problem of responsive flows versus unresponsive flows was first noted by Nagle [4], who introduced a fair bandwidth sharing scheduling algorithm to alleviate this problem. Subsequently, other researchers also realized the importance of providing flow isolation through fair bandwidth sharing and how it can greatly improve the performance of end-to-end congestion control algorithms, resulting in the proposal of many Packet Fair Queuing (PFQ) algorithms [5], [6], [7].

1.2 Towards Quality-of-Service Provisioning

The insufficiencies of the best-effort paradigm have lead to the proposal of other service paradigms, which can be broadly categorized into the two groups of: resource reservation and best-effort enhancements.

1.2.1 Resource Reservation

Paradigms proposed under this category differs from the best-effort paradigm in two fundamental aspects: (1) applications can reserve network resources, like bandwidth,

and (2) the network can accept or reject these reservation requests (also known as admission control) to ensure that a minimum level of service is provided for the reserved traffic.

To provide for these fundamental changes, a plethora of techniques and mechanisms have been developed for packet scheduling, buffer management, admission control, and signaling [8]. These solutions usually require complex signalling and/or state control mechanisms to manage per flow state information, like in Integrated Services (IntServ) [9], or aggregated state information, like in Differentiated Services (DiffServ) [10]. While the proposed solutions are able to provide to a high level of service assurance theoretically, thus resolving the issues of service differentiation and flow isolation, they are not widely deployed because the solutions must be implemented on all the network elements along a flow's path for them to be effective. In reality, this requirement is almost impossible to achieve because a flow will normally traverse across the networks of several operators before reaching its destination, and it is unrealistic to expect all operators to have resource reservation compliant network elements.

1.2.2 Best-Effort Enhancements

Instead of having only a single class of best-effort traffic. Several researchers have proposed to enhance the best-effort service paradigm by having several classes of best-effort traffic, each with a different service priority. Unlike the previous category, service differentiation is achieved without resource reservation signalling or admission control. Therefore, the proposed solutions for best-effort enhancements service models are usu-

ally more scalable and simpler to deploy compared to the solutions of the reservation based service models.

However, these solutions can only provide a relative form of service differentiation and cannot provide any guarantees or flow isolation within each class of best-effort traffic. Proponents do not see this as a big disadvantage because in reality the service guarantees promised by the resource reservation solutions is almost impossible to achieve due to the need to have all network elements resource reservation compliant. On the other hand, enhancements to best-effort traffic can be incrementally deployed, the more routers implementing it, the more effective is the service differentiation.

Another reason for choosing best-effort enhancements over resource reservation is the desire to maintain the flat rate type of commercial agreement between network operators and subscribers. The historical study of communications infrastructure has shown that consumers prefer the simplicity of flat rate pricing and operators offering such pricing tend to experience better demand than those offering usage-based pricing [11].

1.3 Contributions

1.3.1 Thesis Scope and Focus

The broad subject of this thesis is on service differentiation in packet networks. Although most of the contributions are applicable to any packet-based network technology, the communication network platform in focus will be the IP-based Internet platform.

Within the subject of service differentiation, the focus is on providing service differentiation and flow isolation within best-effort traffic, which is more scalable and simpler to deploy than the other reservation based service models because it does not require signalling, admission control, or bandwidth brokers.

1.3.2 Contributions

Having limited the scope and focus of this thesis, an original service differentiation framework, called the **Delay-Rate Differentiated Services (DRDS)**, is proposed.

DRDS consists of two major portions (see Figure 1.1):

(1) The first portion addresses the issue of inefficient resource utilization by providing delay based service differentiation among classes of traffic aggregates. It is based on the Delay-Rate Differentiation (DRD) model, which refines on the Proportional Delay Differentiation (PDD) model, proposed by Dovrolis under the Proportional Differentiated Services (PDS) framework [13]. The DRD model is a combination of the PDD model with another proposed model, called the Generalized Minimum Queueing Delay (GMQD)[1] model [15], [16], [17]. The PDD is a model that provides delay based proportional differentiation among backlogged service classes traversing a single link. The GMQD is a model that minimizes the total queueing delay of all backlogged service classes traversing a single link. Depending on traffic load conditions, DRD is able to switch between PDD and GMQD, thus exploiting the advantages of both models. Two classes of packet scheduling algorithms emulating GMQD and DRD are also proposed

[1] Also known as Minimum Potential Delay in [14].

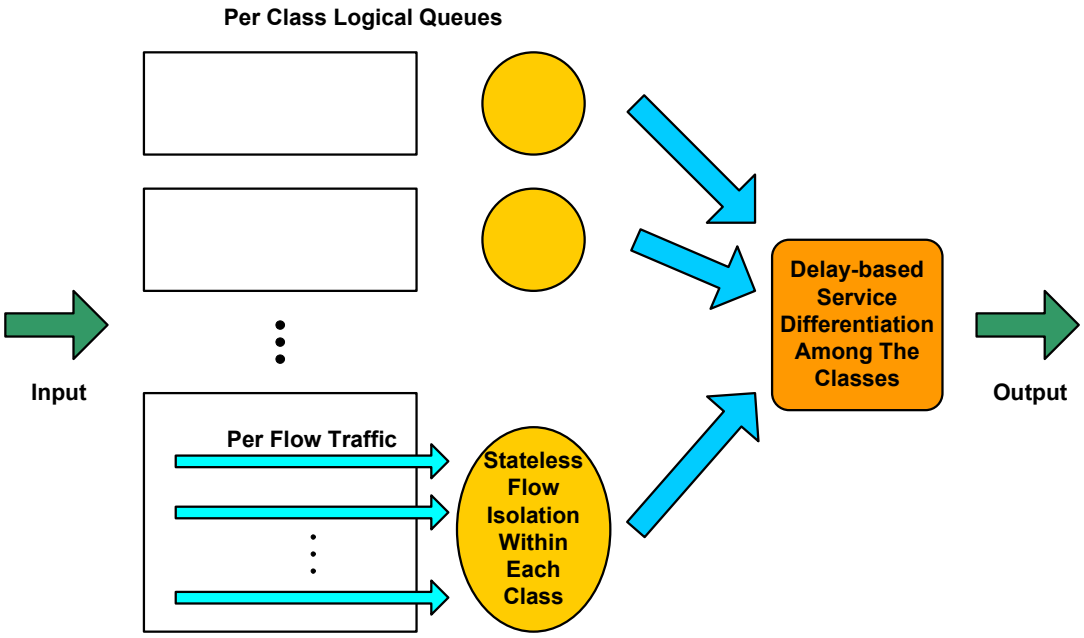

Figure 1.1: Overview of the Delay-Rate Differentiated Services Framework.

and analyzed.

Subsequently, a novel approximation technique that reduces the computational complexity of one class of algorithms proposed for GMQD and DRD is proposed. This technique reduces the computational complexity of the scheduling algorithms without comprising on their performance.

(2) The second portion complements the first portion by addressing the issue of congestion responsive flows versus congestion unresponsive flows within each class of traffic aggregates. A novel control-theoretical approach, which enhances the flow isolation performance of existing fair queueing algorithms that do not maintain per flow state information is proposed.

1.4 Organization

The rest of this thesis is organized as follows. Chapter 2 describes the prominent architectures for achieving Quality of Service (QoS) in IP networks.

In Chapter 3, an original model for providing delay differentiation, called DRD, is proposed. The rationale, components, and packet scheduling algorithms for this model are described and analyzed.

In Chapter 4, a novel approximation technique is proposed to improve the scalability of one class of packet scheduling algorithms proposed in Chapter 3. This technique is able to reduce the computational complexity without comprising on the scheduling performance of the algorithms.

In Chapter 5, the use of a control-theoretical approach that enhances the flow isolation performance of existing core-stateless fair queueing algorithms is proposed and analyzed.

Finally, Chapter 6 summarizes the conclusions of this thesis and ends with directions for future work.

Chapter 2

Background

The best-effort service model currently used by the Internet has functioned well for simple applications like web-browsing, e-mail and file-transfer. However, as the Internet evolves into a global communication infrastructure, more complex service models must be in place to support applications with specific QoS requirements.

In this chapter, the best-known proposals used to improve the best-effort service model of today's Internet: (a) Integrated Services (IntServ) [9], (b) Differentiated Services (DiffServ) [10], (c) Stateless Core (SCORE) [12], and (d) Proportional Differentiated Services (PDS) [13], are presented. IntServ and DiffServ are Internet Engineering Task Force (IETF) recommended standards, while SCORE and PDS are enhancements that can be deployed over a DiffServ network.

This chapter concludes with a discussion on the relationship between the proposed Delay-Rate Differentiated Services (DRDS) framework and the above mentioned frameworks.

2.1 Integrated Services

IntServ [9] is a per-flow based QoS framework that supports applications with delay and bandwidth requirements. In addition to Best-Effort Service, two other service models are defined. They are: (1) Guaranteed Service for applications with fixed delay requirements and (2) Predictive Service for applications with probabilistic delay requirements. In order to achieve QoS guarantees, a signaling protocol for applications to reserve network resources dynamically, called Resource ReServation Protocol (RSVP) [18], was invented. Subsequently, the two service models were renamed as Guaranteed Service and Controlled Load Service in the implementation specifications [19] and [20] respectively.

2.1.1 Resource Reservation Protocol (RSVP)

RSVP [18] uses a receiver-initiated reservation process that can be used for a multi-cast environment. The signaling process is illustrated in Figure 2.1. The flow source sends a PATH Message to the receiver specifying the characteristics of the traffic. As the PATH Message propagates towards the receiver, each router along the way records the path characteristics of the flow. Upon receiving a PATH Message, the receiver responds with a RESV Message to request resources for the flow. Depending on the available network resources, intermediate router along the path can accept or reject the request. If the request is accepted, link bandwidth and buffer space are allocated to the flow and related flow state information will be installed in the router. If the request is rejected, the router will send an error message to the receiver.

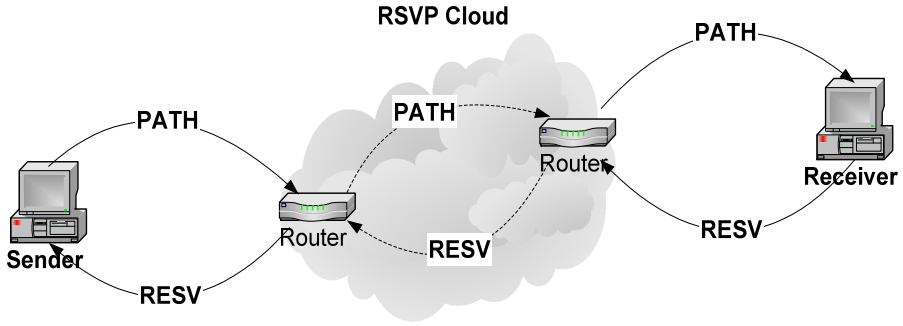

Figure 2.1: RSVP Signalling.

2.1.2 Guaranteed Service

Guaranteed service is aimed to support applications with hard real-time requirements [19]. Under this service model, a flow can be guaranteed a minimum bandwidth. In addition, a maximum end-to-end delay can be computed given the traffic arrival process of the flow. Therefore, Guaranteed Service provides very fine-grained QoS guarantees and is ideal for real-time applications such as IP telephony.

However, the cost associated with Guaranteed Service is the significant increase in complexity. Routers need to maintain per-flow forwarding states and perform per-flow classification, buffer management, scheduling, and admission control. On top of this, resource reservation during admission control is based on worst-case traffic arrival characteristics and this normally leads to significant under-utilization of network resources.

2.1.3 Controlled-Load Service

Controlled-Load service is aimed to support the broad class of adaptive and real-time applications [20]. Under this service model, the packet loss is not significantly larger than

the basic error rate of the transmission medium, and the end-to-end delay experienced by a very large percentage of packets does not greatly exceed the end-to-end propagation delay. The Controlled-Load service is intended to provide better support for a broad class of applications that have been developed for use in today's Internet. Among the applications that fall into this class are video and audio streaming.

The Controlled Load Service trades a lower QoS for a simpler implementation. Although the router still need to perform per-flow admission control, other operations, like packet classification, buffer management, and scheduling can be greatly simplified.

In summary, compared with the current best-effort Internet, IntServ supports a wider range of applications with different QoS requirements. Unfortunately, introducing flow-specific state in routers leads to significant complexity and scalability issues.

2.2 Differentiated Services

To alleviate the complexity issues of Intserv, the Differentiated Services (Diffserv) framework was proposed. The Diffserv architecture differentiates between edge and core routers. Edge routers maintain per-flow state information and perform per-flow operations like buffer management, scheduling and admission control. The assumption is that at the network boundary, there are fewer traffic flows, therefore, edge routers can perform operations at a finer granularity. At the network core, traffic flows are aggregated. Core routers only need to maintain state information for a few classes of aggregated traffic flows. As the number of classes defined is small, packet processing can be efficiently implemented. Hence, this differentiation between edge and core routers

Pool	Code Point Space	Assignment Policy
1	xxxxx0	Standardization
2	xxxx11	Local or Experimental Use
3	xxxx01	Local or Experimental Use (possible standardization)

Table 2.1: Assignment policy of Differentiated Services code points

makes the DiffServ architecture highly scalable.

DiffServ leverages on the relatively unused Type-Of-Service (TOS) byte in IPv4 [21] and Traffic Class byte in IPv6 [22] for Differentiated Services field (DS field) definition [23], [24]. Six bits are used for marking a DiffServ Code Point (DSCP), which provides information about the QoS requested for the packet. Core routers then use this DSCP to classify and select the per-hop behavior (PHB) the packet experiences at each node. The remaining two bits are used for Explicit Congestion Notification (ECN) mechanisms [25].

DiffServ is capable of conveying 64 distinct code points. Presently, the code points are divided into three code point pools, as illustrated in Table 2.1 [23]. The first pool of 32 code points, "xxxxx0", is reserved for standardization. The second pool of 16 code points, "xxxx11", is reserved for local or experimental use. Finally, the third pool of 16 code points, "xxxx01", is initially reserved for local or experimental use, but may be used for standardization purposes in the future if necessary.

14 DSCPs have been defined so far. The best-effort traffic in DiffServ has the default DSCP of 000000. Besides Best-Effort Service, two other service models and their corresponding DSCPs have been defined. They are: (1) Premium Service [26] for applications with low delay requirements and (2) Assured Services [27] for applications

with regular bandwidth requirements.

2.2.1 Premium Service

Premium service is aimed to provide the equivalent of a "virtual leased line" and can be implemented using the Expedited Forwarding (EF) PHB described in [26][1]. The DSCP of EF PHB is 101110. This service model is optimized to provide low delay for applications that generate fixed peak bit-rate. However, end-user must ensure traffic conform to its service-profile. Otherwise, out of profile traffic will be down-graded or dropped. The implementation of Premium service requires admission control, which is handled by a Bandwidth Broker (BB). Each network domain has a BB with complete knowledge about the entire domain. To set up a flow across a domain, the BB must ensure the availability of network resources in its domain before the request is granted. Premium service is suitable for Internet Telephony or for creating virtual lease lines.

Note that Premium service is able to provide different bandwidth requirements for different flows only, unlike Guaranteed service of IntServ that is able to provide different bandwidth and delay requirements for different flows. This is because core routers in DiffServ handles flow aggregates. Therefore, the only way to meet different delay requirements for different flows is to guarantee the smallest delay required by all flows. However, this results in a resource utilization that may be significantly lower than Guaranteed service under IntServ.

[1]Note that this document obsoletes original document described in [28].

	Class 1	**Class 2**	**Class 3**	**Class 4**
Low Drop Precedence	**AF11** 001010	**AF21** 010010	**AF31** 011010	**AF41** 100010
Medium Drop Precedence	**AF12** 001100	**AF22** 010100	**AF32** 011100	**AF42** 100100
High Drop Precedence	**AF13** 001110	**AF23** 010110	**AF33** 011110	**AF43** 100110

Table 2.2: Differentiated Services Code Points of Assured Forwarding Per-Hop Behaviors

2.2.2 Assured Service

Assured service is aimed to provide a certain contracted bandwidth "profile" to the users based on statistical provisioning and is implemented using the Assured Forwarding (AF) PHBs described in [27]. The 12 DSCPs of the AF PHBs are shown in Table 2.2. The AF PHBs provide the delivery of IP packets in four independently forwarded AF classes. Within each AF class, an IP packet can be assigned one of three different levels of drop precedence. A configurable, minimum amount of forwarding resources (buffer space and bandwidth) must be allocated to each implemented AF class. Each AF class may be configured to receive more forwarding resources than the minimum when excess resources are available.

In this service model, user traffic are monitored at the ingress routers and tagged as "In" or "Out" according to their service profiles, which is usually defined in terms of absolute bandwidth and relative loss. Packets are tagged as "In" if the user does not exceeds its service profile and "Out" otherwise. During congestion, "Out" packets are dropped first before "In" packets. Based on this service model, different service

levels such as gold, silver and bronze can be offered. Assured service is suitable for a wide-range of applications, ranging from low delay applications such as adaptive audio streaming to high delay applications such as FTP.

2.2.3 Reconciling Differentiated Services with Integrated Services

In summary, DiffServ scales much better than IntServ because it manages traffic at the aggregate rather than per-flow level. Core routers in the DiffServ region do not distinguish individual flows. They handle packets according to the DiffServ codepoint (DSCP) in the IP header packet, eliminating the need for per-flow state and per-flow processing.

Currently, IntServ and DiffServ are being viewed as complementary technologies for achieving end-to-end QoS [29]. IntServ can be used at the access networks, while DiffServ can be used at the metropolitan area networks (MAN) or wide area networks (WAN). The main benefit of this model is a scalable end-to-end QoS framework, where explicit reservations can be made at the access network. Border routers between the IntServ and DiffServ regions may interact with core routers using aggregate RSVP in the DiffServ region to reserve resources between edges of the region [30].

2.3 Stateless Core

The simplicity of DiffServ is achieved with certain compromises. In order to have the QoS capabilities of IntServ without compromising on the scalability of DiffServ, Stoica proposed the Stateless Core (SCORE) architecture [12].

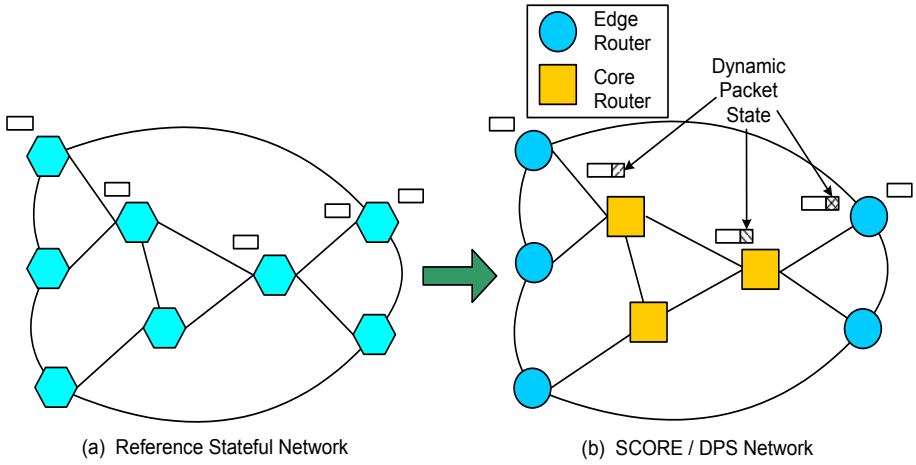

Figure 2.2: (a) A reference stateful network whose functionality is approximated by (b) a Stateless Core (SCORE) network. In SCORE, only edge routers perform per-flow management; core routers do not perform per-flow management. The Dynamic Packet State technique is used to store relevant state information in the packet header so that core routers do not need to maintain per-flow state information.

The goal of SCORE network, as illustrated in Figure 2.2, is to approximate the service of a reference stateful network like IntServ. The key technique used to implement the SCORE network is Dynamic Packet State (DPS). When a packet arrives at the ingress edge router, some state information is inserted into the header of the packet. Core routers process each incoming packet based on the state carried in the header of the packet, updating both its internal state and the state in the header of the packet before forwarding it to the next hop. By using DPS to coordinate actions of edge and core routers along the path traversed by a flow, distributed algorithms can be designed to approximate the behavior of a broad class of stateful networks in which core routers do not maintain per flow state [12].

In [12], Stoica demonstrated how three important Internet services that previ-

ously required stateful network architectures can be implemented using his proposed SCORE/DPS architecture. These services are: guaranteed service, service differentiation for large traffic aggregates, and flow isolation for congestion control.

2.3.1 Guaranteed Service

Stoica demonstrated how a core-stateless version of Jitter Virtual Clock [31] can be implemented using DPS to provide throughput guarantees and end-to-end delay bound without per-flow management. Subsequently, Zhang et. al. in [32], [33], generalized Stoica's scheme to develop a general core-stateless DPS framework called the Virtual Time Reference System (VTRS). A scalable bandwidth broker architecture was also developed based on this VTRS framework [34], [35].

2.3.2 Service Differentiation for Large Traffic Aggregates

Stoica proposed an alternative AF service, Location Independent Resource Accounting (LIRA) [36], in which the service profile is defined in terms of resource tokens rather than fixed bandwidth profiles. Unlike Guaranteed service, LIRA is a form of relative service differentiation that can achieve high network traffic utilization and provide large spatial granularity service, i.e. service assurance can be defined irrespective of where or when a user sends its traffic.

2.3.3 Flow Isolation for Congestion Control

Stoica demonstrated how core-stateless flow isolation using DPS can be achieved for congestion control using his proposed Core-Stateless Fair Queueing (CSFQ) algorithm

[37]. Unlike Guaranteed service and LIRA, this service model does not require any form of resource reservation and can be seen as a form of "Enhanced Best-Effort" service. It forms an important component in the SCORE architecture because the proposed scheduling algorithm can be used to provide flow isolation within the best effort traffic aggregate. This helps to prevent congestion unresponsive flows, like UDP, from squeezing out congestion responsive traffic flows, like TCP, when these two types of traffic are aggregated together in the same service class. Other core-stateless flow isolation algorithms that have been proposed include Rainbow Fair Queueing (RFQ) [38] and Tag-based Unified Fairness (TUF) [39], are proposed.

Note that SCORE/DPS is not an IETF recommended standard, but it can be incrementally deployed over a DiffServ network by network operators using Multi-Protocol Label Switching (MPLS) [12].

2.4 Proportional Differentiated Services

With the exception of the core-stateless flow isolation algorithms proposed under the SCORE architecture, the other service models presented so far in IntServ, DiffServ and SCORE require some form of resource reservation. Recently, providing service differentiation within best-effort traffic has been the object of several contributions, including Proportional DiffServ (PDS) [13], Alternative Best-Effort (ABE) [40], and Best-Effort DiffServ (BEDS) [41]. These frameworks do not require resource reservation, are simpler to implement, and can be used to create a new "Enhanced Best-Effort" service model within the existing DiffServ architecture. Furthermore, network operators

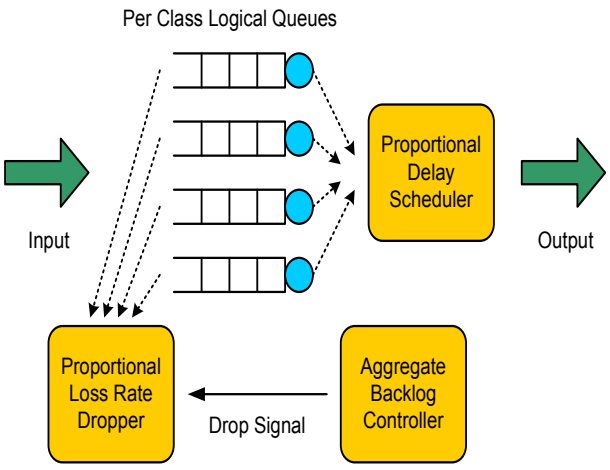

Figure 2.3: The main components of the packet forwarding engine in the Proportional Differentiation Model.

can do flat-rate pricing, which are believed to be the basis for the rapid deployment of Internet [11], for these service models.

One of the framework that has received significant attention is the PDS proposed by Dovrolis [13]. PDS is based on the Proportional Differentiation Model (PDM) [42], which states that class performance metrics based on per-hop queueing delays and packet drops should be proportional to certain differentiation parameters chosen by the network operator [13] (see Figure 2.3). Through these differentiation parameters the network operator can control the relative spacing of the offered classes, based on pricing or policy requirements.

Under this framework are the Proportional Delay Differentiation (PDD) model and the Proportional Loss Differentiation (PLD) model. Schedulers approximating the PDD model that have been studied include Backlog Proportional Rate (BPR) [43], Proportional Queue Control Mechanism (PQCM) [44], Extended Virtual Clock (Ex-VC) [45],

Dynamic Weighted Fair Queueing (DWFQ) [46], and Mean Delay Proportional (MDP) [47]. While droppers approximating the PLD model that have been studied include Proportional Loss Rate (PLR) droppers based on Loss History Table [48] and Average Drop Distances (ADDs) [49].

This framework, however, does not address the issue of flow isolation within each service class.

2.5 Delay-Rate Differentiated Services

Similar to SCORE/DPS and PDS, DRDS is built upon the DiffServ architecture. It consists of two portions that addresses service differentiation and flow isolation independently.

The first portion of the DRDS focuses on providing delay based service differentiation for classes of traffic aggregates. It is based on the Delay-Rate Differentiation (DRD) model, which refines on the Proportional Delay Differentiation (PDD) model, proposed by Dovrolis under the PDS framework. The DRD model is a combination of the PDD model with another proposed model, called the Generalized Minimum Queueing Delay (GMQD) model [15], [16], [17].

In this thesis, packet scheduling algorithms proposed for this portion include Queue Length based Packetized Delay Rate Differentiation (QL-PDRD), Queueing Delay based Packetized Delay Rate Differentiation (QD-PDRD), and an efficient approximation to QD-PDRD called Scaled Queueing Delay based Packetized Delay Rate Differentiation (SQD-PDRD).

The second portion of DRDS focuses on providing flow isolation within each class of traffic aggregates. This portion leverages upon the class of core-stateless fair queueing algorithms proposed under the SCORE/DPS framework. This class of algorithms encodes flow rate information onto the packet using the DPS technique so that core routers, without maintaining per flow state information, are still able to provide approximate fair bandwidth sharing among the flows within each class.

In this thesis, a novel control-theoretical approach is used to enhance CSFQ [37] and RFQ [38], resulting in two new and improved algorithms, called Control-theoretical Approach to CSFQ (CA-CSFQ) [50], [51] and Control-theoretical Approach to RFQ (CA-RFQ) [51].

Chapter 3

Delay-Rate Differentiation Model

This chapter addresses the first portion of the Delay-Rate Differentiated Services (DRDS) framework, which is to provide delay based service differentiation within best-effort traffic so as to utilize network resources more efficiently.

3.1 Background

In this chapter, a new approach for providing delay-based differentiated services, called Delay-Rate Differentiation (DRD), is proposed. DRD refines on the Proportional Delay Differentiation (PDD) model, proposed by Dovrolis under the Proportional Differentiated Services (PDS) framework [13], by combing PDD with another proposed model, called the Generalized Minimum Queueing Delay (GMQD)[1] model [15], [16], [17]. The PDD is a model that provides delay based proportional differentiation among backlogged service classes traversing a single link. While, the GMQD is a model that minimizes the total queueing delay of all backlogged service classes traversing a single link. Depending on traffic load conditions, DRD is able to switch between PDD and GMQD, exploit-

[1] Also known as Minimum Potential Delay in [14].

ing the advantages of both models. Two Packetized-GMQD (PGMQD) algorithms are subsequently proposed and modified to become Packetized-DRD (PDRD) algorithms.

The remainder of this chapter is organized as follows. In Section 3.2, DRD is described in greater details and issues involved in using this class of algorithms are discussed. In Section 3.3, GMQD is described and some of its properties analyzed. This is followed by two proposed Packetized-GMQD (PGMQD) algorithms and their corresponding Packetized-DRD (PDRD) algorithms in Section 3.4 and Section 3.5 respectively. In Section 3.6, simulation results comparing various algorithms are presented. In Section 3.7, other delay-based differentiation models are discussed. Finally, Section 3.8 concludes this chapter.

3.2 PDD, GMQD and DRD

In this section, the differences between PDD and GMQD are analyzed through an analytical fluid flow model. DRD, which is able to exploit certain advantages of PDD and GMQD, is subsequently proposed. Before embarking on the discussion, a summary list of the notations used throughout this chapter is provided in Table 3.1.

Consider a single node system. The total traffic load is r, the capacity is C, and the utilization is $u = r/C < 1$. Assume that the link has adequate buffers to avoid any packet losses. The system offers N classes of service, which are delay differentiated. If $\bar{D}_k$ is the average queueing delay in class k, then the PDD model requires that

$$\frac{\bar{D}_k}{\bar{D}_l} = \frac{\delta_k}{\delta_l} \qquad 1 \leq k, l \leq N \tag{3.1}$$

Notation	Comments
r	total traffic load
C	link capacity
u	link utilization
N	number of service classes in the system
$\bar{D}_k$	average queueing delay in class k
δ_k	delay differentiation parameters of class k
r_k	traffic load in class k
$\bar{Q}_{agg}$	average aggregate backlog
ϕ_k	service rate of class k allocated by the scheduler
w_k	weight assigned to class k
p_k^i	i^{th} packet of service class k
$A(p_k^i)$	time packet p_k^i reaches its head-of-queue
$S(p_k^i)$	start tag of packet p_k^i
$F(p_k^i)$	finish tag of packet p_k^i
L_k^i	length of packet p_k^i
$v(t)$	server virtual time
P_k^i	priority of packet p_k^i
d_k^i	waiting time of packet p_k^i
Δt	packet inter-arrival time

Table 3.1: Notations used in Chapter 3. For simplicity, the notations do not include the time argument t.

where $\delta_1 > \delta_2 > \ldots > \delta_N > 0$ are the Delay Differentiation Parameters (DDPs).

Note that class N is the highest priority class with the lowest average delay and higher priority classes have lower average delays independent of class loads. When the class load distribution $\{r_l\}$ is given, the average queueing delay in class k under the PDD constraints is

$$\bar{D}_k = \frac{\delta_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l} \tag{3.2}$$

where $\bar{Q}_{agg} = \sum_{l=1}^{N} r_l \bar{D}_l$ is the average aggregate backlog in the system. This is the conservation law [52], which states that the $\bar{Q}_{agg}$ is independent of the class load distribution or the scheduling algorithm when the latter is work-conserving and indifferent to packet size. $\bar{Q}_{agg}$ only depends on the link utilization and on the statistical properties of the traffic. Note that the implication of the conservation law is that even though a scheduler can affect the relative magnitude of the class delays, the sum of all $r_l \bar{D}_l$ has to be equal to the average aggregate backlog $\bar{Q}_{agg}$ of the system. This however, does not imply that a scheduler cannot be designed to minimize the sum of all class delays.

Following the above descriptions, if a scheduling algorithm is optimized to minimize the total weighted queueing delay of all backlogged service classes, then the following relationship holds

$$\frac{\bar{D}_k / r_k}{\bar{D}_l / r_l} = \frac{\delta_k}{\delta_l} \qquad 1 \leq k, l \leq N \tag{3.3}$$

This is the GMQD model and the above condition will be derived subsequently during the discussion on the properties of the GMQD model in Section 3.3.

When the class load distribution $\{r_l\}$ is given, the average queueing delay in class k

under the GMQD constraints is

$$\bar{D}_k = \frac{\delta_k r_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l^2} \tag{3.4}$$

where $\bar{Q}_{agg} = \sum_{l=1}^{N} r_l \bar{D}_l$ is the average aggregate backlog in the system.

To better illustrate the differences between PDD and GMQD, consider the example where there are $N = 4$ classes in the system and $\bar{Q}_{agg} = 1$ Mbits. $\delta_4 = 1$ and other δ_k values are set according to $\delta_{k-1}/\delta_k = 2$. Now, if $r_1 = 4$ Mbps, $r_2 = 3$ Mbps, $r_3 = 2$ Mbps, and $r_4 = 1$ Mbps, then the average class delay achieved by GMQD is generally better than PDD (see Table 3.2).

Three observations can be made from this simple example:

(1) Except for the average class delay of the lowest priority class, all the other classes have lower average class delay when using GMQD. As the lowest priority class is the group of users who are paying the least for network usage, deteriorating the average delay of the lowest priority class for the overall improvement in the average delay of other classes is a reasonable alternative.

(2) Overall, the total average class delay in the system is lower for GMQD.

(3) The ratio of delay differentiation, $\bar{D}_{k-1}/\bar{D}_k$, for GMQD has increased. Consistent delay differentiation in the system is not comprised, i.e. higher priority classes experienced delay no worse than lower priority classes.

In general, we can have the following proposition:

Proposition 3.1. *Given $\delta_{k-1}/\delta_k \geq 2$ for $1 < k \leq N$, $\delta_1 = +\infty$ (assuming the lowest priority class is for best effort traffic) and $r_1 > r_2 > \ldots > r_N$, GMQD is able to provide*

k	δ_k	r_k **(Mbps)**	d_k **(msec)**	
			PDD	**GMQD**
1	8	4	163.27	184.97
2	4	3	81.63	69.364
3	2	2	40.82	12.121
4	1	1	20.41	5.7803
Total	15	10	306.12	282.24

Table 3.2: Average class delay for PDD and GMQD when $r_1 = 4$ Mbps, $r_2 = 3$ Mbps, $r_3 = 2$ Mbps, and $r_4 = 1$ Mbps.

lower average class delay for all classes in the system with the exception of the lowest priority class when compared with PDD.

The proof of the above Proposition is given in Appendix A. Note that the underlying assumption for a successful implementation of PDD and GMQD is a pricing scheme that makes higher priority classes more costly (or more usage-restricted) than lower priority classes. Otherwise, everyone will use the highest priority class and the model will not be effective. Past studies on the pricing of interactive computer services [53], [54], suggest that the pricing differences between one service class from another must be substantial, at around a factor of 2. Therefore, network operators are most likely to set $\delta_{k+1}/\delta_k \geq 2$. In addition, the most practical or most frequently experienced scenario based on this pricing model will be $r_1 > r_2 > \ldots > r_N$.

If the class load distribution is equal among the four classes, then the performance of PDD and GMQD are identical.

On the other hand, if the class load is distributed such that $r_1 < r_2 < \ldots < r_N$, then the average class delay of the highest priority class may be longer when using GMQD

k	δ_k	r_k **(Mbps)**	d_k **(msec)**	
			PDD	**GMQD**
1	8	1	307.69	137.93
2	4	2	153.85	137.93
3	2	3	76.92	103.45
4	1	4	38.46	68.966
Total	15	10	576.92	448.28

Table 3.3: Average class delay for PDD and GMQD when $r_1 = 1$ Mbps, $r_2 = 2$ Mbps, $r_3 = 3$ Mbps, and $r_4 = 4$ Mbps.

(see Table 3.3). However, note that the total average class delay in the system remains lower for GMQD.

Summarizing the above discussion. Under a pricing scheme that makes higher priority classes more costly (or more usage-restricted) than lower priority classes, the most frequently experienced traffic scenario will be $r_1 > r_2 > \ldots > r_N$. During such a traffic scenario, GMQD is able to provide lower average class delay than PDD for all, except the lowest priority class. In addition, consistent delay differentiation is also not comprised. However, during other traffic conditions, GMQD may not be able to provide consistent delay differentiation, unlike PDD.

In view of the above, the DRD model is proposed. For DRD, differentiation is based on

$$\frac{\bar{D}_k/\phi_k}{\bar{D}_l/\phi_l} = \frac{\delta_k}{\delta_l} \qquad 1 \leq k, l \leq N \tag{3.5}$$

where ϕ_l is the service rate of class l allocated by the scheduler.

When the class load distribution $\{r_l\}$ is given, the average queueing delay in class k

under the DRD constraints is

$$\bar{D}_k = \frac{\delta_k \phi_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l \phi_l r_l} \tag{3.6}$$

Note that DRD is equivalent to PDD when $\phi_1 = \phi_2 = \ldots = \phi_N$, and equivalent to GMQD when $\phi_l = r_l$ for all classes in the system.

To enjoy the best of both worlds, DRD must be able to provide: (a) consistent delay differentiation among classes independent of the class load distribution and (b) lower average class delay than PDD for all, except the lowest priority class independent of traffic conditions. Following the same derivation used in Appendix A, the above criterias can be met by setting $\phi_k = max(r_k, \phi_{k+1})$, where $\phi_{N+1} = 0$ and ϕ_k is computed sequentially from the highest priority class (class N) to the lowest priority class (class 1).

In the next section, GMQD is discussed in greater detail. This is followed by 2 proposed Packetized-GMQD algorithms in Section 3.4, before deriving their corresponding Packetized-DRD algorithms in Section 3.5.

3.3 Generalized Minimum Queueing Delay

In this section, Generalized Minimum Queuing Delay (GMQD) is described in detail. GMQD can be defined as an idealized algorithm that minimize the total queuing delay of all backlogged service classes traversing a single switching node.

3.3.1 Fluid GMQD Model

GMQD is a work-conserving server that may operate at a fixed or variable rate $C(t)$. Let $\phi_k(t)$ and $Q_k(t)$ be the service rate and backlog of class k at time t respectively, where $k = 1, 2, \ldots, N$. Therefore, a GMQD server serving N classes is defined as one where the sum of weighted queueing delay

$$\sum_{k=1}^{N} \frac{Q_k(t)w_k}{\phi_k(t)} \tag{3.7}$$

is minimized by varying the ϕ_k values at time t, where w_k is the weight assigned to class k.

As the server is work-conserving, the sum of total service rates is always given by

$$\sum_{k=1}^{N} \phi_k(t) = C(t) \tag{3.8}$$

at time t.

Theorem 3.1. *The service rate $\phi_k(t)$ of service class k in a GMQD server at time t is given by*

$$\phi_k(t) = \frac{\sqrt{Q_k(t)w_k}}{\sum_{l=1}^{N} \sqrt{Q_l(t)w_l}} C(t) \tag{3.9}$$

where $Q_k(t)$ is the amount of class k backlog in bits at time t, w_k is the weight assigned to class k and $C(t)$ is the bandwidth of the server at time t.

The proof derived using Dynamic Programming (DP) [55] can be found in Appendix B[2]. This theorem gives the exact service rates of each class required to minimize the

[2]Alternatively, the Lagrange Multiplier [56] method can be used to derive the proof.

total weighted queueing delay. However, for implementation purposes, only a relative value is required for the computation of ϕ_l.

Corollary 3.1. *A GMQD server can be implemented by varying ϕ_k according to*

$$\phi_k(t) = \sqrt{Q_k(t) w_k} \tag{3.10}$$

where $Q_k(t)$ is the amount of flow k backlogged in bits at time t and w_k is a real number denoting the weight assigned to class k.

Proof: From Theorem 3.1, divide the service rates of classes k and l to obtain

$$\frac{\phi_k}{\phi_l} = \sqrt{\frac{Q_k w_k}{Q_l w_l}},$$

$$\phi_k = K\sqrt{Q_k w_k}, \tag{3.11}$$

where $K = \phi_l / \sqrt{Q_l w_l}$ is a constant of proportionality that every ϕ_k can be expressed in terms of, for $k = 1, 2, \ldots, N$. As the ϕ_k value for every class k in the system can be expressed in terms of the same K multiplied by its $\sqrt{Q_k w_k}$ value, it is sufficient to use relative values in the implementation of a rate based scheduler. The proof completes.

□

Note that the expression in Corollary 3.1 does not require any information from the other classes. This result will be used in Section 3.4 in the design of a packet emulation of GMQD called Queue Length based Packetized GMQD (QL-PGMQD).

3.3.2 Heavy Traffic Conditions

The performance of GMQD under limiting heavy traffic conditions (see equation (3.3)), i.e. when the total traffic load $r = \sum_{k=1}^{N} r_k$ tends towards the link capacity C, is derived.

Theorem 3.2. *Given an N service classes GMQD system with a work-conserving server operating at fixed rate C. When the total traffic load conforms to* $\sum_{k=1}^{N} r_k = C$*, the average backlog* $\bar{Q}_k$ *and average queueing delay* $\bar{D}_k$ *eventually converges to:*

$$\bar{Q}_k = \frac{\bar{Q}_{agg}}{\sum_{l=1}^{N} \left(\frac{w_k}{w_l}\right) \left(\frac{\bar{r}_l}{\bar{r}_k}\right)^2} \tag{3.12}$$

$$\bar{D}_k = \frac{\bar{Q}_{agg}}{\bar{r}_k \sum_{l=1}^{N} \left(\frac{w_k}{w_l}\right) \left(\frac{\bar{r}_l}{\bar{r}_k}\right)^2}. \tag{3.13}$$

where $\bar{Q}_{agg} = \sum_{l=1}^{N} r_l \bar{D}_l$ *is the average aggregate backlog in the system.*

The proof of the above theorem can be found in Appendix C.

Dividing the average queueing delay for two classes, resulting in

$$\frac{\bar{D}_k/\bar{r}_k}{\bar{D}_l/\bar{r}_l} = \frac{w_l}{w_k} \tag{3.14}$$

where the DDPs ratio is the inverse of the weights ratio, i.e.

$$\frac{\delta_k}{\delta_l} = \frac{w_l}{w_k} \tag{3.15}$$

Note that combining equation (3.14) and (3.15) will result in equation (3.3).

An alternative implementation of PGMQD based on the use of average queueing delay will be proposed in Section 3.4.2.

3.4 Packetized Generalized Minimum Queueing Delay

GMQD is an idealized discipline that does not transmit packets as entities. It assumes that the server can serve multiple service classes simultaneously and that the traffic is infinitely divisible. In this section, packet emulations of GMQD are studied. The first

one is similar to Packet Fair Queueing (PFQ) algorithms [5], [6], [7], [57], [58], while the next is similar to the Time Dependent Priorities (TDP) [59] scheme, also known as Waiting Time Priorities (WTP) [43].

3.4.1 Queue Length based Packetized Generalized Minimum Queueing Delay

The main idea behind Queue Length based Packetized Generalized Minimum Queueing Delay (QL-PGMQD) is to use the Generalized Processor Sharing (GPS) model [5], [60], with the modification that the service rate of each service class is proportional to the square root of the weighted backlog of the service class (see Corollary 3.1). To better estimate the normalized amount of service time received by each class, the average normalized service rates value for every $[A(p_k^i), A(p_k^{i+1})]$ time interval is used. Specifically, the proposed QL-PGMQD algorithm is defined as:

(1) The i^{th} packet, p_k^i of service class k on reaching the head of a First-In-First-Out (FIFO) queue at time $A(p_k^i)$ is stamped with a start tag $S(p_k^i)$ and a finish tag, $F(p_k^i)$, which are determined as:

$$S(p_k^i) = max(F(p_k^{i-1}), v(A(p_k^i))) \tag{3.16}$$

$$F(p_k^i) = S(p_k^i) + \frac{2L_k^i}{\sqrt{Q_k(A(p_k^i))w_k} + \sqrt{Q_k(A(p_k^{i+1}))w_k}} \tag{3.17}$$

where L_k^i is the length of p_k^i, w_k is the weight of class k, $Q_k(t)$ is the number of bits backlogged in class k at time t and $S(p_k^0) = 0$.

(2) The server virtual time $v(t)$ is initially set to zero. During a busy period, $v(t)$ is

defined to be equal to the start tag of the packet in service at time t.

(3) Packets are serviced in increasing order of their start tags. Any ties are broken arbitrarily.

(4) At the end of a busy period, the server virtual time, $v(t)$ together with the start and finish tags of every class are re-initialized to zero.

Note that scheduling for QL-PGMQD is based on the "smallest start tag first" policy because the finish tag $F(p_k^i)$ can only be computed at $A(p_k^{i+1})$ (refer to equation (3.17)).

In terms of implementation complexity, one source of complexity for QL-PGMQD is the sorted priority list operations used to locate the packet with the smallest start tag. Depending on the sorted list algorithm used, the complexity can range from $O(log_2 N)$ for an exact sorted priority list implementation to $O(1)$ for an approximate implementation [61], [62].

The other source of complexity lies in the need to do a floating point square root operation for tag computation. A commercially available Pentium III processor takes around 40 clock cycles and 80 clock cycles to execute a double precision floating point division and double precision extended square root operation respectively. Hence, it can be concluded that the time required for executing one QL-PGMQD tag computation (involving one floating point division and one square root) is around three times the amount of time required for executing one tag computation of a conventional fair queueing algorithm (involving one floating point division) such as Start-Time Fair Queueing (SFQ) [63], [64]. Note that the overhead associated with one floating point addition/subtraction is small compared with one floating point division (or square root)

because one addition/subtraction operation can be issued every clock cycle.

3.4.2 Queueing Delay based Packetized Generalized Minimum Queueing Delay

An alternative implementation of GMQD is to schedule packets based on priorities computed using the delay expression derived in Theorem 3.2. Specifically, Queueing Delay based Packetized Generalized Minimum Queueing Delay (QD-PGMQD) is defined as follows:

(1) The priority of the i^{th} packet, p_k^i of service class k is computed based on

$$P_k^i = \frac{d_k^i w_k}{r_k} \tag{3.18}$$

where P_k^i and d_k^i is the priority and waiting time of packet p_k^i respectively. w_k is the weight assigned to class k and r_k is the exponential moving average rate of class k computed using:

$$r_k^{new} = (1 - e^{-\Delta t/\tau})/\Delta t + e^{-\Delta t/\tau} r_k^{old} \tag{3.19}$$

where Δt is the packet inter-arrival time, and τ is a constant[3].

(2) Packets are serviced in increasing order of their priority tags. Any ties are broken arbitrarily.

For QD-PGMQD, the priority tags of all the head of queue packets have to be computed each time a new packet reaches its head of the queue leading to a computational complexity of $O(N)$, where N is the number of service classes in the system. This may

[3]A good discussion on what value to set for τ can be found in [12].

be difficult to implement for systems with many classes but feasible for DiffServ if we assume eight classes or less [13].

3.5 Packetized Delay Rate Differentiation

In this section, two Packetized DRD (PDRD) algorithms, called Queue Length based Packetized Delay Rate Differentiation (QL-PDRD) and Queueing Delay based Packetized Delay Rate Differentiation (QD-PDRD), are proposed. Both PDRD algorithms are extensions of the two PGMQD algorithms proposed in Section 3.4.

3.5.1 Queue Length based Packetized Delay Rate Differentiation

The main concepts behind Queue Length based Packetized Delay Rate Differentiation (QL-PDRD) is based on QL-PGMQD. Specifically, the proposed QL-PDRD algorithm is defined as:

(1) The i^{th} packet, p_k^i of service class k on reaching the head of a First-In-First-Out (FIFO) queue at time $A(p_k^i)$ is stamped with a start tag $S(p_k^i)$ and a finish tag, $F(p_k^i)$, which are determined as:

$$S(p_k^i) = max(F(p_k^{i-1}), v(A(p_k^i))) \tag{3.20}$$

$$F(p_k^i) = S(p_k^i) + \frac{2L_k^i}{\phi_k(A(p_k^i)) + \phi_k(A(p_k^{i+1}))} \tag{3.21}$$

where

$$\phi_k(t) = \max(\sqrt{Q_k(t)w_k}, \sqrt{Q_{k+1}(t)w_{k+1}}) \tag{3.22}$$

L_k^i is the length of p_k^i. w_k is the weight of class k. $\phi_k(t)$ is the service rate allocated to class k at time t. $Q_k(t)$ is the number of bits backlogged in class k at time t and $S(p_k^0) = 0$. Note that the $max(.)$ term in equation (3.22) is used to account for the $r_{k-1} \leq r_k$ traffic condition and $Q_{N+1}(t)w_{N+1} = 0$.

(2) The server virtual time $v(t)$ is initially set to zero. During a busy period, $v(t)$ is defined to be equal to the start tag of the packet in service at time t.

(3) Packets are serviced in increasing order of their start tags. Any ties are broken arbitrarily.

(4) At the end of a busy period, the system's virtual time, $v(t)$ together with the start and finish tags of every class are re-initialized to zero.

Once again, the scheduling for QL-PDRD is based on the "smallest start tag first" policy because the finish tag $F(p_k^i)$ can only be computed at $A(p_k^{i+1})$ (refer to equation (3.21)).

The implementation complexity is in the same order as QL-PGMQD, which are limited by the sorted priority list operations used to locate the packet with the smallest start tag and the square root computation.

3.5.2 Queueing Delay based Packetized Delay Rate Differentiation

An alternative implementation of DRD is to schedule packets based on priorities computed using packet delay. Specifically, Queueing Delay based Packetized Delay Rate Differentiation (QD-PDRD) is defined as follows:

(1) The priority of the i^{th} packet, p_k^i of service class k is computed sequentially from

class $k = N$ to class $k = 1$ based on

$$P_k^i = \frac{d_k^i w_k}{\phi_k} \tag{3.23}$$

where

$$\phi_k = \max(r_k, \phi_{k+1}) \tag{3.24}$$

and $\phi_{N+1} = 0$. P_k^i and d_k^i is the priority and waiting time of packet p_k^i respectively. w_k is the weight assigned to class k. ϕ_k is the service rate assigned to class k and r_k is the exponential moving average rate of class k, computed using the same equation as equation (3.19).

(2) Packets are serviced in increasing order of their priority tags. Any ties are broken arbitrarily.

The complexity of QD-PDRD is $O(N)$, which is in the same order as QD-PGMQD and WTP, because the priority tags of all the head of queue packets have to be computed each time a new packet reaches its head of the queue.

3.6 Simulation Results

For purpose of comparison, the simulation scenarios and parameters are similar to that of [43].

3.6.1 Single Node

In this section, simulations are used to evaluate the performance of BPR, WTP (TDP), QL-PGMQD, QD-PGMQD, QL-PDRD and QD-PDRD from a single node perspective.

The effects of different class load distributions on the long term average delay differentiation and the performance of each algorithm transiting from different class load distributions are investigated.

The simulation scenario is a scheduler that is loaded with traffic from $N = 4$ sources. In all cases, packet inter-arrivals follow a Pareto distribution with a shape parameter $\alpha = 1.9$. The packet length distribution is the same for all classes, where 50% of the packets are 40 bytes, 35% are 550 bytes, and 15% are 1500 bytes, giving a good representative of the majority of the packets seen in the Internet [65]. The bandwidth of the network links is 51.84 Mbps. The link utilization $u = r/C$ used is 0.9. The weights of each class will be chosen as $w_k/w_{k-1} = 2$.

Note that the class of DRD algorithms is based on the assumption that $\delta_1 = +\infty$. If δ_1 is not significantly larger than other δ terms, then based on the derivation used in Appendix A an additional $\phi_1 = max\left[max(r_1, \phi_2), \phi_2 + \frac{\sum_{l=3}^{N} \delta_l r_l (\phi_2 - \phi_l)}{\delta_1 r_1}\right]$ condition is required. This additional condition is used in this simulation scenario because $\delta_1 = 8$ is not significantly larger than $\delta_2 = 4$.

Figure 3.1 shows the average class delays using BPR, QL-PGMQD, QL-PDRD, WTP, QD-PGMQD, and QD-PDRD for five different class load distribution. Each point in these figures is obtained from averaging over ten simulation runs with different random number generator seeds, each run being of 100 sec duration. The results are segregated into two groups: Figure 3.1(a) shows the results of the class of "Queue Length" based algorithms (BPR, QL-PGMQD, and QL-PDRD), while Figure 3.1(b) shows the results of the class of "Queueing Delay" based algorithms (WTP, QD-PGMQD, and

QD-PDRD).

From the results, it can be seen that similar to the BPR and WTP algorithms, the QL-PDRD and QD-PDRD algorithms are able to achieve consistent delay differentiation, where higher priority classes consistently achieve better delay than lower priority classes independent of the five different load distributions. Furthermore, the QL-PDRD and QD-PDRD algorithms are able to provide lower average class delay for all classes with the exception of the lowest priority class (i.e. class 1) when compared with the BPR and WTP algorithms respectively. This is achieved independent of the five different load distributions and without significant deterioration of the average delay of the lowest priority class. Therefore, these results showed that DRD is able to maintain the consistent delay differentiation characteristics of PDD, while exploiting the total delay minimization characteristics of GMQD to provide better overall average class delays than PDD.

Figure 3.2 shows the ratio of the average delays between successive classes using the same set of results. The results are segregated into three groups: Figure 3.2(a) shows the results of the PDD algorithms, Figure 3.2(b) shows the results of the GMQD algorithms, and Figure 3.2(c) shows the results of the DRD algorithms. For comparison, three columns of data points are plotted for each class load distribution: the first column of points denote the results of the ideal fluid flow models, the second column of points denote the results of the class of "Queueing Delay" based algorithms, and the last column of points denote the results of the class of "Queue Length" based algorithms.

From the results, it can be seen that the class of "Queueing Delay" based algorithms,

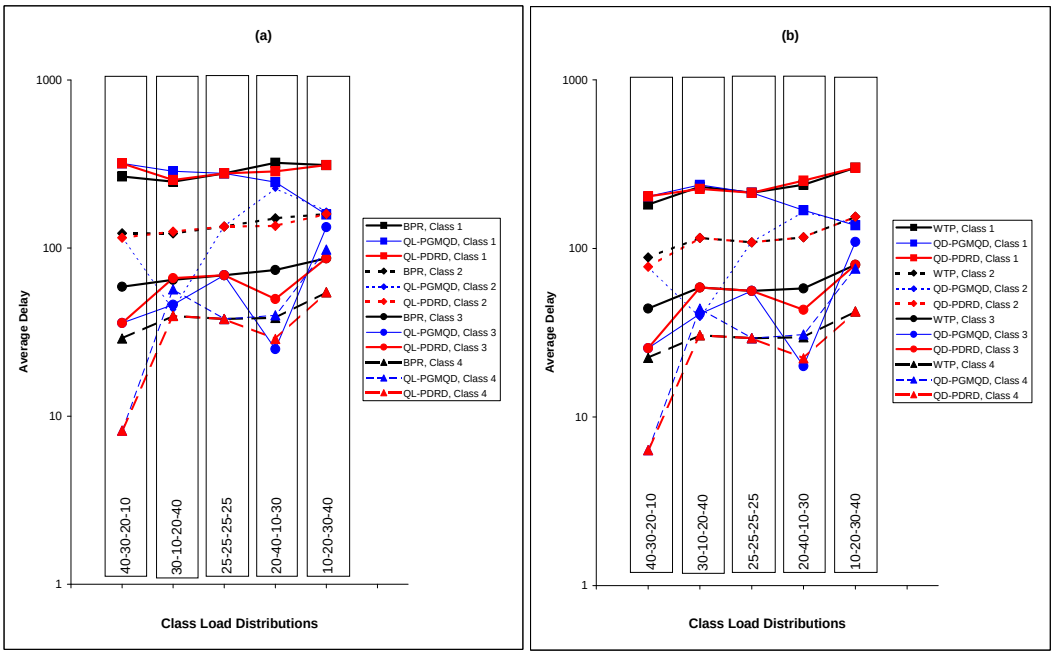

Figure 3.1: The average class delays using BPR, QL-PGMQD, QL-PDRD, WTP, QD-PGMQD, and QD-PDRD for different class load distribution. The four numbers in each bar denote the fraction of the four classes in the aggregate packet stream, starting from class 1 up to class 4. The link utilization is 90%. (a) The simulation results using BPR, QL-PGMQD, and QL-PDRD. (b) The simulation results for WTP, QD-PGMQD, and QD-PDRD.

such as WTP, QD-PGMQD, and QD-PDRD, is less dependent on the load distribution and is able to achieve the delay differentiation ratios close to the values specified by the ideal fluid flow models. On the other hand, the class of "Queue Length" based algorithms, such as BPR, QL-PGMQD, and QL-PDRD, is more dependent on the load distribution. These results indicate that the class of "Queueing Delay" based algorithms have better performance than the class of "Queue Length" based algorithms. This better performance, however, comes at the expense of greater computational complexity.

3.6.2 Multiple Nodes

In this experiment, the focus is on the end-to-end performance of the packet flows. The issue here is whether local class-based relative differentiation can lead to consistent end-to-end flow-based relative differentiation. This set of simulations will only evaluate the six algorithms for the weights ratio $w_k/w_{k-1} = 2$. Figure 3.3 shows the network topology, which is a typical multiple congested nodes network configuration. The number of nodes varies from 1 to 5. The input traffic at the first node is generated by $N = 4$ sources and traverses all nodes in the network configuration. The traffic characteristics and class load distribution is the same as that of the previous experiments. At each node, cross-traffic that are generated from $P = 4$ sources are included. Their traffic characteristics and class load distribution follows that of the input traffic sources. The link utilization is $u = 90\%$ with the aggregate cross-traffic load taking up 70% of the total traffic at each link. The bandwidth of the network links is 51.84 Mbps. In order to examine the effectiveness of the relative delay differentiation, only the queueing de-

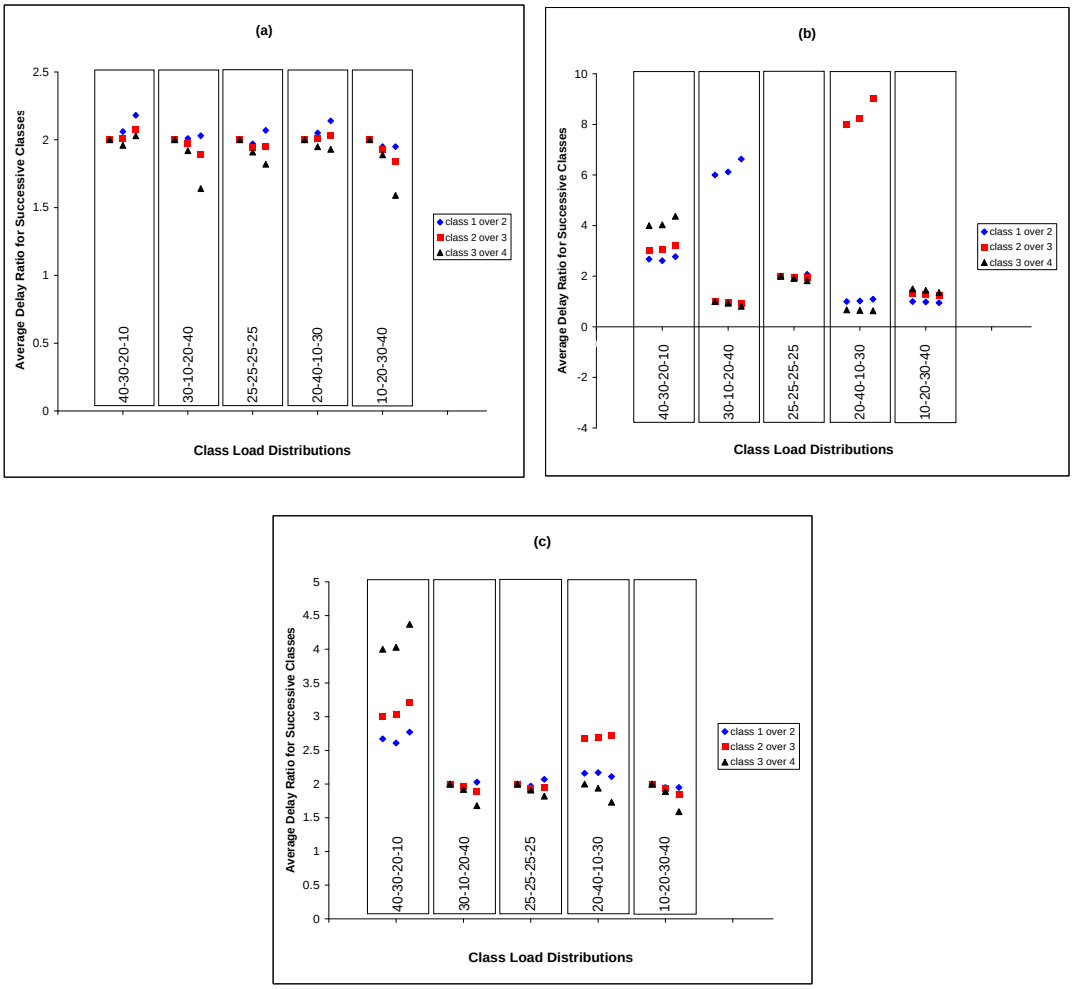

Figure 3.2: The ratio of average delays between successive classes with different class load distribution. The four numbers in each bar denote the fraction of the four classes in the aggregate packet stream, starting from class 1 up to class 4. (a) The three columns of points in each bar denote, from left to right, the simulation results for PDD, WTP, and BPR respectively. (b) The three columns of points in each bar denote, from left to right, the simulation results for GMQD, QD-PGMQD, and QL-PGMQD respectively. (c) The three columns of points in each bar denote, from left to right, the simulation results for DRD, QD-PDRD, and QL-PDRD respectively.

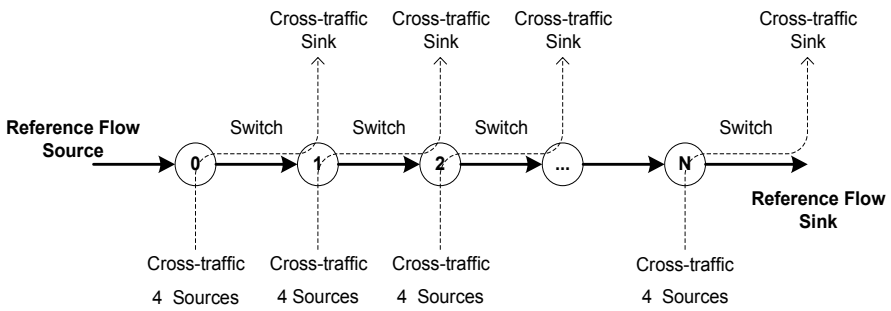

Figure 3.3: Multiple congested link network configuration.

lays are considered. Propagation and transmission delays are ignored because they are common to all packets.

Figure 3.4 shows the ratio of the average delays between successive classes in multiple congested nodes network configuration. The class load distribution is chosen as $r_1 = r_2 = r_3 = r_4 = 0.25r$. From the results of Figure 3.4(a), the performance of BPR, QL-PGMQD and QL-PDRD are comparable. Similarly, from the results of Figure 3.4(b), the delay performance of WTP, QD-PGMQD and QD-PDRD are also comparable. Another observation is that for all six algorithms, the deviations from the weights ratio reduces as the number of nodes that the input traffic traverse increases. Therefore, this study shows that local class-based relative differentiation leads to consistent end-to-end flow-based relative differentiation.

3.7 Related Work

Other relative differentiation models that differ slightly from PDM have also been studied and proposed. Bodamer [66] extends on PDD to propose a scheduler that pro-

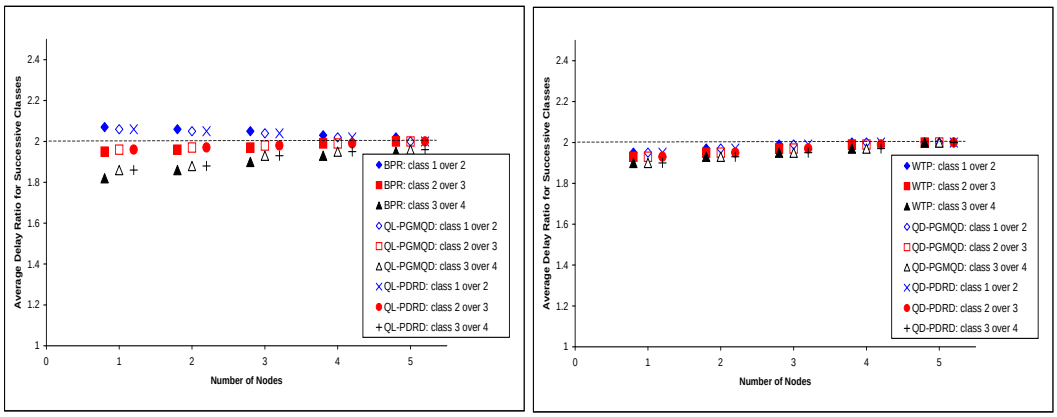

Figure 3.4: (a) The ratio of average-delays between successive classes using BPR, QL-PGMQD and QL-PDRD for class load distribution of $r_1 = r_2 = r_3 = r_4 = 0.25r$. The weights ratio is $w_k/w_{k-1} = 2$. (b) The same simulation using WTP, QD-PGMQD and QD-PDRD.

vides proportional delay violation probabilities among classes in the system. Another interesting framework is proposed by Christin and Liebeherr, called Joint Buffer Management and Scheduling (JoBS) [67]. JoBS combines both link scheduling and queue management in a single algorithm. Furthermore, it is able to offer relative and absolute guarantees on both delay and loss. Heuristics are used to modify service rates so that all QoS constraints can be met. If this is not possible, then packets are dropped. Recently, the same authors also presented a quantitative assured forwarding service based on linear feedback control theory [68]. Unfortunately, these alternative models come at the price of additional complexity.

3.8 Conclusion

This chapter presents a new model for providing delay based differentiation, called Delay-Rate Differentiation (DRD) model. DRD is based on PDD and GMQD. It exploits the advantage of PDD and GMQD by taking on the characteristics of either model depending on traffic load conditions. Two classes of packet scheduling algorithms are subsequently proposed to emulate GMQD and DRD, one based on "Queue Length" and the other based on "Queueing Delay".

Chapter 4

Achieving Delay Differentiation Efficiently

In Chapter 3, it was demonstrated that the class of Queueing Delay based packet scheduling algorithms has better performance compared to the class of Queue Length based packet scheduling algorithms, but comes at the expense of computational complexity. In this chapter, a novel approximation technique is proposed to improve the scalability of this class of Queueing Delay based packet scheduling algorithms. This reduction in complexity is achieved without comprising on the scheduling performance of the algorithms.

4.1 Background

Under the framework of DiffServ, Dovrolis presented the Proportional Differentiation Model (PDM) [42] and found the Waiting Time Priority (WTP) scheme [43] to be a suitable scheduling algorithm to achieve proportional delay differentiation (PDD). In the WTP algorithm, the service priority of a packet in queue i at time t is given by $p_i(t) = d_i(t)w_i$, where $d_i(t)$ is the waiting time of the packet at time t and w_i is the

weight of the delay class. The complexity of WTP scheduler is $O(N)$, where N is the number of service classes in the system, because the priority for every backlogged class has to be calculated each time a new packet reaches its head-of-queue. In addition, tag comparisons among all backlogged classes are required to identify the highest priority packet to schedule.

The main contribution of this paper is our proposed Scaled Time Priority (STP) scheme [69], [70], [71], which is an efficient approximation to WTP. Specifically, STP is able to provide near proportional delay at a complexity of $O(1)$, which is lower than the $O(N)$ complexity of WTP. The advantage of STP over WTP is most significant and apparent in a high-speed system with many service classes. DiffServ uses six bits of the IPv4 or IPv6 header to convey the Differentiated Services Codepoints (DSCPs), which selects a Per-Hop Behavior (PHB) [23], [24]. This translates to a maximum of $2^6 = 64$ possible DSCPs. Presently, only 14 DSCPs have been defined: one for Default Best Effort traffic [23], twelve for Assured Forwarding traffic [27], and one for Expedited Forwarding traffic [26]. Network operators, therefore, have the freedom to customize the other DSCPs to meet their own requirements. With WTP, providing PDD for a large number of classes is difficult due to its $O(N)$ complexity. In [13], N was assumed to be eight or less, making the associated implementation complexity surmountable. However, with the simplicity of STP, network operators can have greater freedom to deploy fine granularity PDD over a greater number of service classes.

Another important contribution of this chapter is the derivations of the workload that must be transmitted before an arbitrary packet for WTP and STP schedulers,

where workload is defined as the transmission time of all traffic in the scheduler that must be transmitted before a certain packet of interest. The derived workload allows one to quantify the exact schedulability conditions of a scheduling algorithm given the traffic arrival characteristics [61], [62]. In the past, the $O(1)$ RPQ+ algorithm used a similar workload derivation to establish that it can efficiently approximate the $O(log_2 N)$ complexity Earliest Deadline First (EDF) algorithm [72]. In this chapter, the derived workload for the WTP scheduler will be used as the reference for providing PDD. The derived workload for the $O(1)$ complexity STP scheduler is then compared with $O(N)$ complexity WTP to establish that both algorithms are comparable.

The rest of the chapter is organized as follows. In Section 4.2, PDM is discussed in greater detail. In Section 4.3, WTP is described and the workload transmitted before an arbitrary tagged packet using the WTP scheduler is derived. In Section 4.4, the proposed STP algorithm is described and the workload transmitted before an arbitrary tagged packet using a STP scheduler is derived. Subsequently, the differences between WTP and STP are reconcilied. In Section 4.5, simulations are used to evaluate the algorithms. In Section 4.6, the same method used in simplifying WTP is applied to Queueing Delay based Packetized Generalized Minimum Queueing Delay (QD-PGMQD) and Queueing Delay based Packetized Delay Rate Differentiation (QD-PDRD). In Section 4.7, STP is compared with other related work, before concluding in Section 4.8.

Before embarking on the discussion, a summary list of the notations used throughout this chapter is provided in Table 4.1.

Notation	Comments
d_i	average waiting time of departed (non-dropped) packets for class i
δ_i	delay differentiation parameters of class i
l_i	average packet loss rate for class i
σ_i	loss differentiation parameters of class i
w_i	scheduler differentiation parameter of class i or the weight assigned to class i
p_i	service priority of class i
N	number of service classes in the system
P	number of disjoint priority classes in the system
C_i	priority class i
$A_j[t, t+\tau]$	total session j traffic that arrives to the scheduler in the time interval $[t, t+\tau]$
t_p	time tagged packet from session j in class C_p arrives at the scheduler
$t_p + \delta_p$	time tagged packet from session j in class C_p starts to depart from the scheduler
$t_p + \tau_p$	arbitrary time between t and $t_p + \delta_p$
B_j	backlog in the scheduler from session j
$t_p + \hat{\tau}_p$	last time before t_p that the scheduler does not contain packets that are to be transmitted before the tagged packet
$R(t_p + \hat{\tau}_p)$	remaining transmission time of packet transmitted at time $t_p + \hat{\tau}_p$
D	parameter used to scale the waiting time of packet in the Scaled Time Priority algorithm
T	parameter used to scale the time the packet reaches its head-of-queue in the Scaled Time Priority algorithm
$t_p + h_p$	time tagged packet from session j in class C_p reaches its head-of-queue
Δ_{q-p}	time difference between packets from priority classes q and p reaching their respective head-of-queues
ϕ_i	service rate of class i allocated by the scheduler
Δt	packet inter-arrival time

Table 4.1: Notations used in Chapter 4. For simplicity, the notations do not include the time argument t.

4.2 Proportional Differentiation Model

There are several ways to provide relative differentiated services. For example, the differentiation can be strictly based on pricing (like Paris Metro Pricing [53]), or on capacity provisioning (like Weighted Fair Queueing and its variants [5], [6], [7], [57], [58], [64], [73]). However, these mechanisms cannot provide consistent differentiation between classes especially for traffic patterns that are bursty over a wide range of timescales. Other mechanisms like strict prioritization can provide consistent class differentiation that does not depend on load variations, but do not allow network operators to adjust the quality spacing between classes.

In view of the above, Dovrolis proposed the Proportional Differentiation Model (PDM) [42]. PDM has two key features:

(1) Predictability, in the sense that the differentiation is consistent (i.e. higher classes are better, or at least no worse) and independent of the variations of the class loads, and

(2) Controllability, meaning that the network operators are able to adjust the quality spacing between classes based on their selected criteria.

The PDM states that certain class performance metrics, related to the per-hop queueing delays and packet drops, should be proportional to the corresponding differentiation parameters that the network operator chooses [13]. Through these differentiation parameters the network operator can control the relative spacing (ratio) of the offered classes, based on pricing or policy requirements. Specifically, let d_i be the average queueing delay of the departed (non-dropped) packets. The proportional delay differentiation

(PDD) model requires that the class average delays are spaced as

$$\frac{d_i}{d_j} = \frac{\delta_i}{\delta_j} \quad 1 \leq i, j \leq N \tag{4.1}$$

The parameters δ_i are the Delay Differentiation Parameters (DDPs), and they are ordered so that higher classes provide lower delays, i.e., $\delta_1 > \delta_2 > \ldots > \delta_N > 0$. Schedulers approximating the PDD model that have been studied include Backlog Proportional Rate (BPR) [43], Proportional Queue Control Mechanism (PQCM) [44], Extended Virtual Clock (Ex-VC) [45], Dynamic Weighted Fair Queueing (DWFQ) [46], and Mean Delay Proportional (MDP) [47]. Providing PDD efficiently is the focus of our study in this chapter.

Similar to equation (4.1), let l_i be the average loss rate for class i, that is the long-term fraction of class i packets that have been dropped. The PDM in the case of packet drops requires that class loss rates are spaced as

$$\frac{l_i}{l_j} = \frac{\sigma_i}{\sigma_j} \quad 1 \leq i, j \leq N \tag{4.2}$$

The parameters σ_i are the Loss rate Differentiation Parameters (LDPs), and they are also ordered as $\sigma_1 > \sigma_2 > \ldots > \sigma_N > 0$. Droppers approximating the PLD model that have been studied include Proportional Loss Rate (PLR) droppers based on Loss History Table [48] and Average Drop Distances (ADDs) [49]. PLD will not be considered in this thesis.

Note that the PDM must be strongly coupled with a pricing or a policy-based scheme to make higher classes more costly (or more usage-restricted) than lower classes. Otherwise, everyone will use the highest class and the model will not be effective.

4.3 Waiting Time Priority

4.3.1 Algorithm

The Waiting Time Priority (WTP) algorithm was first studied by L. Kleinrock in 1964 [59], with the name Time Dependent Priorities (TDP). It was later used by Dovrolis as an effective means to achieve the PDD model. WTP is a priority scheduler in which the priority of a packet increases proportionally with its waiting-time. The algorithm for WTP can be defined as:

(1) All packets on arrival at the scheduler is timestamped with an arrival tag equal to its arrival time.

(2) Whenever a packet reaches its head-of-queue, the priority for every backlogged queue i at time t has to be computed as

$$p_i(t) = d_i(t)w_i \tag{4.3}$$

where $d_i(t)$ is the waiting-time of the packet at time t and w_i is the Scheduler Differentiation Parameter (SDP) that determines the rate with which the priority of the packets of a certain class increases with time.

(3) The scheduler then chooses the packet with the largest priority tag for service. Ties are arbitrarily broken.

From the description above, it is clear that the complexity of WTP is $O(N)$, where N is the number of classes in the system. This is because the priority for every backlogged class has to be calculated each time a new packet reaches its head-of-queue.

In addition, it can be shown that under heavy traffic conditions, the DDP ratios (defined in Section 4.2) tends to the inverse of the corresponding SDP ratios [43], that is

$$\frac{d_i}{d_j} \rightarrow \frac{\delta_i}{\delta_j} = \frac{w_j}{w_i} \quad 1 \leq i, j \leq N \tag{4.4}$$

Next, WTP is analyzed in greater detail and the workload that must be transmitted before an arbitrary packet for a WTP scheduler is derived. This derived workload allows one to quantify the exact schedulability conditions of WTP and will be used as the reference for providing PDD. The derivation follows that of [61], [62], which uses a general traffic arrival characteristics. A similar analysis was done by Kleinrock assuming Poisson arrivals to derive the average delay of classes in a TDP (or WTP) system [52]. Dovrolis later extended on this work to show that proportional delay can be achieved for Poisson arrivals under heavy traffic conditions [13].

4.3.2 Workload that must be Transmitted before an Arbitrary Packet for a Waiting Time Priority Scheduler

The scheduler is assumed to be empty at time 0 and the transmission rate of the scheduler is normalized for simplification of notation. Sessions submitting traffic to a WTP scheduler are further assumed to be grouped into P disjoint priority classes C_1, C_2, ..., C_P, arranged in ascending priority order, where C_P has a higher priority than C_{P-1} and so on.

For a given session j in class C_p, let $A_j[t, t+\tau]$ denote the total session j traffic that arrives to the scheduler in time interval $[t, t+\tau]$ measured in terms of transmission time

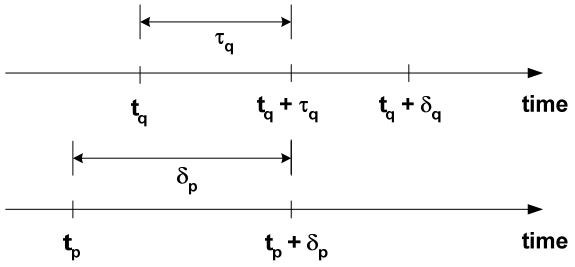

Figure 4.1: Time-lines of a tagged packet p and another arbitrary packet q in a WTP system.

of the link. Note that no assumption is made regarding the traffic arrival characteristics for $A_j[t, t+\tau]$. Besides Poisson arrivals, other traffic models [5], [72], [74], can also be substituted into $A_j[t, t+\tau]$ to derive different workload conditions.

Without loss of generality, assume that a tagged packet from session j in class C_p arrives to a WTP scheduler at time t_p and starts to depart from the scheduler at time $t_p + \delta_p$. Defining $t_p + \tau_p$ as an arbitrary time between t_p and $t_p + \delta_p$, where the tagged packet is in the queue of class C_p. The expression $W^{p,t_p}(t_p + \tau_p)$, which represents the total transmission time of all traffic in the scheduler at time $t_p + \tau_p$ that must be transmitted before the tagged packet can depart, will be derived.

Consider an arbitrary class C_q and determine the time intervals for which packet arrivals from sessions $j \in C_q$ have higher priority than the tagged packet. Figure 4.1 shows an illustration of the time-lines of a tagged packet p and another arbitrary packet q in a WTP system. We start by first assuming that the transmission of a packet can be preempted at any time by a packet arrival with higher precedence. Two cases have to be considered: sessions from the same class as the tagged packet ($q = p$) and sessions

of different priority classes from the tagged packet ($q \neq p$).

(a) $\mathbf{q = p}$: Since all packets from class C_p are transmitted in First-In-First-Out (FIFO) order, all packets from sessions in C_p that arrive in the time interval $[0, t_p]$ will be transmitted before the tagged packet.

(b) $\mathbf{q \neq p}$: For a different priority class C_q with $q \neq p$, the packets transmitted before the tagged packet are those that satisfy the following condition:

$$\tau_q w_q > \delta_p w_p \tag{4.5}$$

where τ_q is the amount of time spent by an arbitrary packet in the queue of class C_q.

Note that the time the tagged packet starts to depart from the scheduler $t_p + \delta_p$ is also the last time tag computations and comparisons are made involving the tagged packet. Therefore, $t_q + \tau_q = t_p + \delta_p$, implying that

$$t_q < t_p + \delta_p - \delta_p \frac{w_p}{w_q}$$

Consequently, all packets from sessions in C_q that arrive in the time interval $[0, t_p + (1 - \frac{w_p}{w_q})\delta_p]$ will be transmitted before the tagged packet.

Note also that for higher (lower) priority class C_q with $q > p$ ($q < p$) and $w_q > w_p$ ($w_q < w_p$), the resulting time interval is greater (smaller) than $[0, t_p]$, the corresponding time interval for C_p. In addition, $q = p$ implies that $w_q = w_p$. Therefore, the time interval $[0, t_p + (1 - \frac{w_p}{w_q})\delta_p]$ is the general form for both cases of $q = p$ and $q \neq p$.

The interval shown above describe the traffic transmitted before the tagged packet, but these intervals assume that the transmission of a packet can be interrupted and preempted. In particular, consider a scenario where at some time prior to the arrival

of the tagged packet at time t_p, there are no packets in the scheduler with arrival times included in the intervals described above. Since the WTP scheduler is work-conserving, some packets which are not included in the intervals described above may be transmitted before the tagged packet. Next, such a non-preemption is accounted for in order to accurately quantify the traffic to be transmitted before the tagged packet.

Define $t_p - \hat{\tau_p}$ to be the last time before t_p that the WTP scheduler does not contain packets that are to be transmitted before the tagged packet. Note that such a time is guaranteed to exist since the scheduler is assumed to be empty at time 0. If $B_j(t)$ is used to denote the backlog in the WTP scheduler from session $j \in C_q$ at time t, then $\hat{\tau_p}$ can be written directly from the intervals above as follows:

$$\begin{aligned} \hat{\tau_p} &= min\{z \mid \sum_{q=1}^{p} \sum_{j \in C_q} B_j(t_p - z) + \\ &\quad \sum_{q=p+1}^{P} \sum_{j \in C_q} B_j(min\{t_p - z, t_p + (1 - \frac{w_p}{w_q})\delta_p\}) = 0, z \geq 0\} \end{aligned} \tag{4.6}$$

Note that $min\{t_p - z, t_p + (1 - \frac{w_p}{w_q})\delta_p\}$ is considered for packets from sessions in priority classes greater than p because $t_p + (1 - \frac{w_p}{w_q})\delta_p$ is smaller than t_p.

By definition of time $t_p - \hat{\tau_p}$, the traffic transmitted by the WTP scheduler during the interval $[t_p - \hat{\tau_p},\ t_p + \delta_p]$ is limited to packets with arrival times during the intervals specified above and the remaining transmission time of some other packet in transmission at time $t_p - \hat{\tau_p}$, which is denoted by $R(t_p - \hat{\tau_p})$.

Finally, an expression for $W^{p,t_p}(t_p + \tau_p)$ can be written, i.e. the workload in the scheduler at time $t_p + \tau_p$ that will be transmitted before the tagged packet from class

C_p with arrival time t_p is completely transmitted. This expression is given as follows: For all τ_p, $0 \leq \tau_p \leq \delta_p$,

$$W^{p,t_p}(t_p + \tau_p) = \sum_{q=1}^{P} \sum_{j \in C_q} A_j[t_p - \hat{\tau_p}, t_p + (1 - \frac{w_p}{w_q})\delta_p] + R(t_p - \hat{\tau_p}) - (\hat{\tau_p} + \tau_p) \quad (4.7)$$

The first term on the right hand side of equation (4.7) accounts for the arrival intervals of all P disjoint priority classes in the system that were derived previously, while the term $R(t_p - \hat{\tau_p})$ is the remaining transmission time of the packet transmitted at time $t_p - \hat{\tau_p}$. Since by choice of $\hat{\tau_p}$, the packet scheduler is continuously backlogged for the entire interval $[t_p - \hat{\tau_p}, t_p + \tau_p]$, the final term accounts for the total workload transmitted during the interval.

4.4 Scaled Time Priority

In this section, the Scaled Time Priority (STP) scheduler is introduced and the workload transmitted before an arbitrary packet for this scheme is derived. The expression derived above for the WTP algorithm will then be compared with that of STP, so that an evaluation of the two schemes can be made. The intuition behind STP and a discussion on its implementation complexity will also be provided.

4.4.1 Algorithm

The proposed Scaled Time Priority (STP) algorithm can be described as follows:

(1) Each packet on arrival at the scheduler is timestamped with an arrival tag equal to

its arrival time.

(2) A packet on reaching its head-of-queue i at time t has its priority computed as:

$$p_i(t) = \frac{d_i(t)w_i + D}{t + T} \tag{4.8}$$

where $d_i(t)$ is the waiting-time of the packet at time t, w_i is the Scheduler Differentiation Parameter (SDP) that determines the rate with which the priority of the packets of a certain class increases with time, D and T are two parameters used to scale the waiting-time of the packet and time the packet reaches its head-of-queue respectively. It will be shown later in Section 4.4.3 that the condition for STP to approximate WTP can be reduced to:

$$\min \left| \sum_{i=1}^{P} (\frac{Tw_i}{D} - 1) \right| \tag{4.9}$$

where T and D are chosen to satisfy the $T >> t_i + h_i$ and $D >> h_i w_i$ conditions respectively. t_i is the arrival time of the packet to the STP scheduler and $t_i + h_i$ is the time the packet reaches its corresponding head-of-queue.

(3) The scheduler then chooses the packet with the largest priority tag for service. Ties are broken arbitrarily.

(4) Timer of the scheduler resets $t = 0$ at the end of each busy period so that $p_i(t)$ will not gradually tend to zero as t keeps increasing.

From the above description, it is clear that the computational complexity of STP is $O(1)$ because only the new packet that reaches its head-of-queue requires priority computation. The priorities of other head-of-queue packets that had arrived before this new packet do not require re-computation. This is the key difference between STP and

WTP because for WTP the priorities of all head-of-queue packets in the system needs to be re-computed each time a new packet reaches its head-of-queue.

The intuition behind equation (4.8) of STP may not be clear, in Section 4.4.4, it will be shown why having t in the denominator of equation (4.8) alone, without parameters T and W, is sufficient to provide approximate WTP performance. Subsequently, it will be shown why including parameters T and W lead to performance enhancements.

4.4.2 Workload that must be Transmitted before an Arbitrary Packet for Scaled Time Priority Scheduler

All the assumptions laid down in the Section 4.3.2 remain valid. In addition, it is assumed that the tagged packet from session j in class C_p reaches its corresponding head-of-queue at time $t_p + h_p$, where $t_p + h_p$ is a value that lies between its arrival time t_p and its departure time $t_p + \delta_p$. Figure 4.2 shows the time-lines of a tagged packet p and another arbitrary packet q in a STP system, where

$$\Delta_{q-p} = t_q + h_q - (t_p + h_p) \tag{4.10}$$

Δ_{q-p} is the time difference between packets from priority classes q and p reaching their respective head-of-queues. In a STP system, this can also be interpreted as the time difference between priority calculations of packets from priority classes q and p. However, this interpretation is not applicable in a WTP system because there is no time difference between priority calculations of different classes, i.e. priorities of all head-of-queue packets in the system needs to be re-computed each time a new packet reaches its head-of-queue. Note that $t_p + h_p + \Delta_{q-p}$ is limited by $t_p + \delta_p$ because priority

comparisons are only meaningful when the tagged packet is still within the system.

Following the same approach as that described in Section 4.3.2, we first assume that the transmission of a packet can be preempted at any time by a packet arrival with higher precedence. Once again, we consider the two cases under STP: sessions from the same class as the tagged packet ($q = p$) and sessions of different priority classes from tagged packet ($q \neq p$).

(a) $\mathbf{q = p}$: Since all packets from class C_p are transmitted in FIFO order, all packets from sessions in C_p that arrive in the time interval $[0, t_p]$ will be transmitted before the tagged packet.

(b) $\mathbf{q \neq p}$: In a STP system, a packet q that reaches its corresponding head-of-queue at time $t_q + h_q$ will have a priority given by (see equation (4.8)):

$$\frac{h_q w_q + D}{t_q + h_q + T}$$

Note that the amount of time spent waiting in the scheduler by packet q $d_q(t)$ at the time of priority computation, i.e. $t = t_q + h_q$, is always equal to h_q for a STP scheduler because packet q is assumed to arrive at time t_q. The priority of the tagged packet takes a similar form. Therefore, for a priority class C_q with $q \neq p$, packets transmitted before the tagged packet are those that satisfy the following condition:

$$\frac{h_q w_q + D}{t_q + h_q + T} > \frac{h_p w_p + D}{t_p + h_p + T} \tag{4.11}$$

From equation (4.10):

$$t_q = t_p + h_p + \Delta_{q-p} - h_q$$

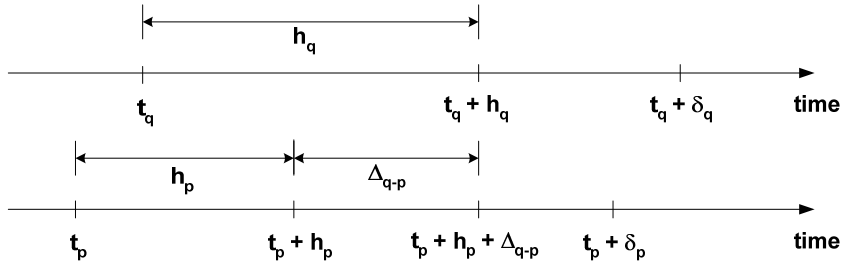

Figure 4.2: Time-lines of a tagged packet p and another arbitrary packet q in a STP system.

$$
\begin{aligned}
&< \quad t_p + (h_p + \Delta_{q-p}) - \left[(\frac{t_q + h_q + T}{t_p + h_p + T})(\frac{h_p w_p + D}{w_q}) - \frac{D}{w_q} \right] \\
&= \quad t_p + (h_p + \Delta_{q-p}) - (1 + \frac{\Delta_{q-p}}{t_p + h_p + T})(\frac{h_p w_p + D}{w_q}) + \frac{D}{w_q} \\
&= \quad t_p + (1 - \frac{w_p}{w_q})(h_p + \Delta_{q-p}) + \frac{w_p}{w_q}(h_p + \Delta_{q-p}) - \\
&\qquad\qquad (1 + \frac{\Delta_{q-p}}{t_p + h_p + T})(\frac{h_p w_p + D}{w_q}) + \frac{D}{w_q} \\
&= \quad t_p + (1 - \frac{w_p}{w_q})(h_p + \Delta_{q-p}) + \frac{w_p}{w_q}\Delta_{q-p} - \frac{\Delta_{q-p}}{w_q}(\frac{h_p w_p + D}{t_p + h_p + T}) \\
&= \quad t_p + (1 - \frac{w_p}{w_q})(h_p + \Delta_{q-p}) + \frac{w_p}{w_q}\Delta_{q-p}(\frac{t_p + T - D/w_p}{t_p + h_p + T}) \qquad (4.12)
\end{aligned}
$$

Since $h_p + \Delta_{q-p} \leq \delta_p$, therefore, all packets from sessions in C_q that arrive in the time interval $[0, t_p + (1 - \frac{w_p}{w_q})\delta_p + \frac{w_p}{w_q}\Delta_{q-p}(\frac{t_p+T-D/w_p}{t_p+h_p+T})]$ will be transmitted before the tagged packet.

Again, note that for higher (lower) priority class C_q with $q > p$ ($q < p$) and $w_q > w_p$ ($w_q < w_p$), the resulting time interval is greater (smaller) than $[0, t_p]$, the corresponding time interval for C_p. In addition, $q = p$ implies that $w_q = w_p$ and $\Delta_{q-p} = 0$. Therefore, the time interval $[0, t_p + (1 - \frac{w_p}{w_q})\delta_p + \frac{w_p}{w_q}\Delta_{q-p}(\frac{t_p+T-D/w_p}{t_p+h_p+T})]$ is the general form for both cases of $q = p$ and $q \neq p$.

Next, the earlier assumption that the transmission of a packet can be interrupted and preempted is accounted for in order to accurately quantify the traffic to be transmitted before the tagged packet. Again defining $t_p - \hat{\tau_p}$ to be the last time before t_p that the STP scheduler does not contain packets that are to be transmitted before the tagged packet and $B_j(t)$ as the backlog in the STP scheduler from session $j \in C_q$ at time t, then:

$$\begin{aligned}\hat{\tau_p} &= min\{z \mid \sum_{q=1}^{p}\sum_{j\in C_q} B_j(t_p - z) + \\ &\quad \sum_{q=p+1}^{P}\sum_{j\in C_q} B_j(min\{t_p - z, t_p + (1 - \frac{w_p}{w_q})\delta_p + \\ &\quad \frac{w_p}{w_q}\Delta_{q-p}(\frac{t_p + T - D/w_p}{t_p + h_p + T})\}) = 0, z \geq 0\} \end{aligned} \tag{4.13}$$

Denoting $R(t_p - \hat{\tau_p})$ as the remaining transmission time of some other packet in transmission at time $t_p - \hat{\tau_p}$, the expression for $W^{p,t_p}(t_p + \tau_p)$, can now be written as:

For all τ_p, $0 \leq \tau_p \leq \delta_p$,

$$\begin{aligned}W^{p,t_p}(t_p + \tau_p) &= \sum_{q=1}^{P}\sum_{j\in C_q} A_j[t_p - \hat{\tau_p}, \\ &\quad t_p + (1 - \frac{w_p}{w_q})\delta_p + \frac{w_p}{w_q}\Delta_{q-p}(\frac{t_p + T - D/w_p}{t_p + h_p + T})] + \\ &\quad R(t_p - \hat{\tau_p}) - (\hat{\tau_p} + \tau_p) \end{aligned} \tag{4.14}$$

The first term on the right hand side of equation (4.14) accounts for the arrival intervals of all P disjoint priority classes in the system that were derived previously, while the $R(t_p - \hat{\tau_p})$ term is the remaining transmission time of the packet transmitted at time

$t_p - \hat{\tau_p}$. Since by choice of $\hat{\tau_p}$, the packet scheduler is continuously backlogged for the entire interval $[t_p - \hat{\tau_p}, t_p + \tau_p]$, the final term accounts for the total workload transmitted during the interval.

4.4.3 Reconciliation between STP and WTP

Taking $W^{p,t_p}(t_p + \tau_p)$ derived for WTP as the reference (see equation (4.7)), we can see that the performance of STP comes closest to WTP when the arrival intervals of STP

$$\sum_{q=1}^{P} \sum_{j \in C_q} A_j[t_p - \hat{\tau_p}, t_p + (1 - \frac{w_p}{w_q})\delta_p + \frac{w_p}{w_q}\Delta_{q-p}(1 - \frac{h_p + D/w_p}{t_p + h_p + T})]$$

tend towards the arrival intervals of WTP

$$\sum_{q=1}^{P} \sum_{j \in C_q} A_j[t_p - \hat{\tau_p}, t_p + (1 - \frac{w_p}{w_q})\delta_p]$$

Note that $\frac{w_p}{w_q}\Delta_{q-p}(1 - \frac{h_p+D/w_p}{t_p+h_p+T})$ can either be a positive or negative value. Therefore, for STP to come closest to WTP, we need to minimize this term for all P classes in the system:

$$min \left| \sum_{p=1}^{P} \sum_{q=1}^{P} \frac{w_p}{w_q}\Delta_{q-p}(1 - \frac{h_p + D/w_p}{t_p + h_p + T}) \right| \tag{4.15}$$

This condition can be further simplified by choosing parameters T and D to satisfy $T >> t_p + h_p$ and $D >> h_p w_p$ respectively:

$$min \left| \sum_{q=1}^{P} \sum_{p=1}^{P} \frac{\Delta_{q-p}}{w_q} w_p(1 - \frac{D}{T w_p}) \right| \tag{4.16}$$

Note that the resulting D/T ratio ensures that the values chosen for both D and T can be large because for a particular end result, D will always be a multiple of the T value chosen.

Further note that Δ_{q-p} can be replaced by a worst-case $|\Delta_{q-p}|_{max}$ term. From equation (4.10),

$$\begin{aligned} |\Delta_{q-p}|_{max} &= |t_q + h_q|_{max} - |t_p + h_p|_{min} \\ &= |t_q + h_q|_{max} \\ &< \frac{h_q w_q T}{D} \end{aligned} \tag{4.17}$$

Based on the initial assumption that the scheduler is empty at time 0 (see Section 4.3.2), both $(t_p + h_p)$ and $(t_q + h_q)$ must be equal or greater than zero. A possible $|t_p + h_p|_{min} = 0$ scenario occurs when a tagged packet p arrives at time $t_p = 0$ and reaches its head-of-queue at time $t_p + h_p = 0$. Based on this scenario, the maximum $(t_q + h_q)$ value that an arbitrary packet q can possess, for it to be still transmitted before the tagged packet p, can be derived from equation (4.11).

Substituting equation (4.17) into equation (4.16),

$$\min \left| \sum_{q=1}^{P} h_q \sum_{p=1}^{P} (\frac{T w_p}{D} - 1) \right| \tag{4.18}$$

Note that the worst-case h_q is restricted by $(B - 2L_{min})/C$, where B is the buffer of the scheduler, L_{min} is the minimum packet length that the tagged packet p and the arbitrary packet q can possess, and C is the link capacity.

As the value of h_q cannot be affected by the design parameters T and D, the condition for STP to approximate WTP reduces to:

$$\min \left| \sum_{p=1}^{P} (\frac{T w_p}{D} - 1) \right| \tag{4.19}$$

This is the same condition described earlier in the STP algorithm (see Section 4.4.1).

In Section 4.5, specific implementations making use of equation (4.19) are shown. Extensive simulations are then conducted to show that STP is a good emulation of WTP.

4.4.4 Discussion

In this section, we will provide additional insights into the STP algorithm. We start by showing how having t in the denominator of equation (4.8) alone without the parameters T and D is sufficient to provide approximate WTP performance. Next, we will show why including parameters T and D leads to performance enhancements in terms of approximating WTP more closely.

The effects of having t in the denominator of equation (4.8) alone without the parameters T and D can be easily derived from the workload expression of equation (4.14). Note that this is equivalent to having $T = 0$ and $D = 0$, resulting in:

$$\begin{aligned} W^{p,t_p}(t_p + \tau_p) &= \sum_{q=1}^{P} \sum_{j \in C_q} A_j[t_p - \hat{\tau_p}, \\ &\qquad t_p + (1 - \frac{w_p}{w_q})\delta_p + \frac{w_p}{w_q}\Delta_{q-p}(\frac{t_p}{t_p + h_p})] + \\ &\qquad R(t_p - \hat{\tau_p}) - (\hat{\tau_p} + \tau_p) \end{aligned} \tag{4.20}$$

From equation (4.20), we can see that WTP performance is achieved when $t_p = 0$ and a good emulation can also be achieved when $t_p << h_p$. Unfortunately, the value of t_p will be significant under bursty traffic conditions with long busy periods. Assuming a positive Δ_{q-p} value, the case where $T = 0$ and $D = 0$ will result in greater workload for all classes in the system (see Figure 4.3(a)).

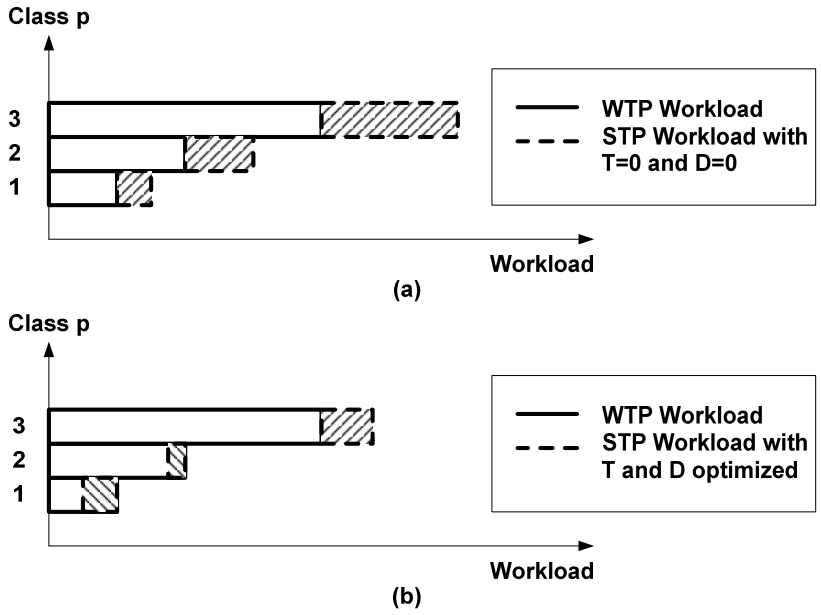

Figure 4.3: An example of STP illustrating the effects of parameters T and D on the workload of each class: (a) when $T = 0$ and $D = 0$, (b) when T and D are optimized.

On the other hand, when T and D are optimized using equation (4.19), the workload deviations between WTP and STP are minimized for all classes in the system (see Figure 4.3(b)), thereby making the performance of STP closer to WTP. Note that unlike the case where $T = 0$ and $D = 0$, the workload of lower priority classes will be smaller compared to WTP, thus equalizing the larger workload of higher priority classes.

4.4.5 Implementation Complexity

The scheduling complexity of an algorithm consists of two key components:

(1) the complexity involved in computing the priority tags, and

(2) the complexity involved in identifying the highest priority head-of-queue packet to service.

For WTP, the tag computation component has a complexity of $O(N)$ because the priority for every backlogged class has to be calculated each time a new packet reaches its head-of-queue. Identifying the highest priority head-of-queue packet to schedule requires $N-1$ comparisons because all N priority tags have different values after each round of tag computations. Therefore, the overall complexity of WTP is $O(N)$.

For STP, the tag computation component has a complexity of $O(1)$, which is a reduction in complexity when compared with WTP. Identifying the highest priority head-of-queue packet to schedule can also be implemented with $O(1)$ complexity in the following two ways:

(1) If the number of service classes is small, then parallel hardware comparators can be used to identify the highest priority head-of-queue packet to schedule;

(2) Alternatively, a $O(1)$ complexity approximate sorted priority queue, like RPQ+ [62], can be used.

Note that DiffServ implies that we are dealing with traffic aggregates making approximate scheduling acceptable. Therefore, the performance advantage from using a $O(log_2 N)$ complexity exact sorted priority queue is marginal over a $O(1)$ complexity approximate sorted priority queue. Hence, we can conclude that the overall complexity of STP is $O(1)$.

DiffServ uses six bits of the IPv4 Type-of-Service or the IPv6 Traffic Class octet to convey the DSCPs, which selects a PHB [23], [24]. This translates to a maximum of $2^6 = 64$ possible DSCPs. Presently, only 14 DSCPs have been defined: one for Best Effort traffic [23], twelve for Assured Forwarding traffic [27], and one for Expedited

Forwarding traffic [26]. Network operators, therefore, have the freedom to customize the other DSCPs to meet their own requirements. For example, network operators can deploy fine granularity PDD by setting SDPs according to $s_{i+1}/s_1 = 2i$ (where $1 \leq i < N$ and $s_1 = 1$) instead of $s_i/s_{i-1} = 2$, resulting in $N = 32$ for $s_N = 62$. With WTP, this implementation is difficult due to its $O(N)$ complexity. Hence, the simplicity of the $O(1)$ complexity STP gives it the scalability unmatched by WTP, providing network operators greater implementation freedom to deploy fine granularity PDD over a greater number of service classes.

4.5 Simulation

4.5.1 Single Node

In this section, extensive simulations similar to [43] are used to evaluate how closely STP emulates WTP from a single node perspective. The effects of different link utilization by aggregate load and of different class load distributions on the long term average delay differentiation are first investigated. The performance of WTP and STP in short timescales are investigated subsequently.

The simulation scenario is a WTP/STP scheduler that is loaded with traffic from $P = 4$ sources. In all cases, packet inter-arrival times follow a Pareto distribution with a shape parameter $\alpha = 1.9$. The packet length distribution is the same for all classes, where 50% of the packets are 40 bytes, 35% are 550 bytes, and 15% are 1500 bytes, giving a good representation of the majority of the packets seen in the Internet [65].

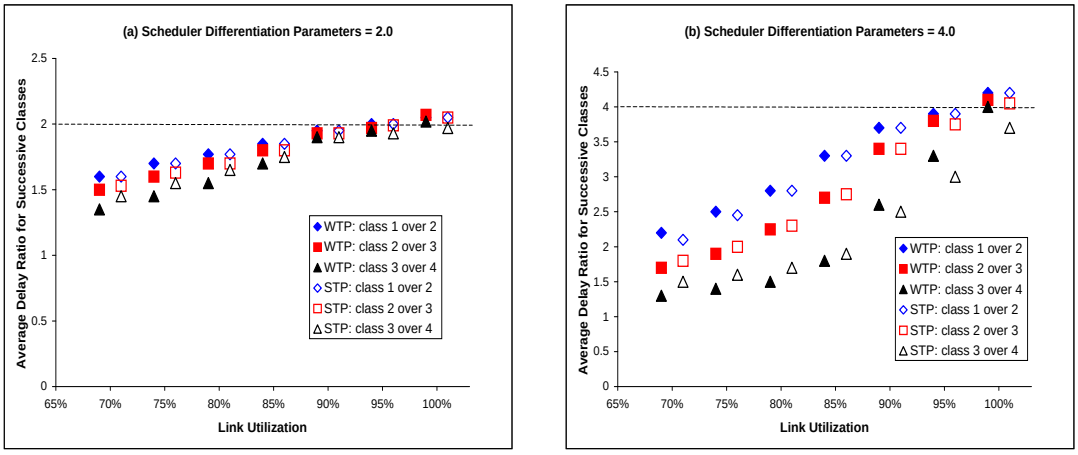

Figure 4.4: The ratio of average-delays between successive classes with WTP and STP for different link utilizations. The traffic load distribution is Class-1: 40%, Class-2: 30%, Class-3: 20%, Class-4: 10%.

The class load distribution in most cases is set to: Class 1: 40%, Class 2: 30%, Class 3: 20%, Class 4: 10%, unless otherwise stated. The bandwidth of the network links is 51.84 Mbps. SDPs will be chosen such that $w_i/w_{i-1} = 2$ or $w_i/w_{i-1} = 4$. To minimize $\sum_{p=1}^{P}\sum_{q=1}^{P}\frac{w_p}{w_q}(1-\frac{D}{Tw_p})$, (1) $D = 375$ sec and $T = 100$ sec for SDPs chosen such that $w_i/w_{i-1} = 2$, and (2) $D = 2125$ sec and $T = 100$ sec for SDPs chosen such that $w_i/w_{i-1} = 4$.

Link Utilization by Aggregate Load

Figure 4.4 shows the ratio of the average delays between successive classes in moderate and heavy load conditions. Each point in these figures is obtained from averaging over ten simulation runs with different random number generator seeds, with each run being of 100 sec duration. The SDPs were chosen such that $w_i/w_{i-1} = 2$ for Figure 4.4(a)

and $w_i/w_{i-1} = 4$ for Figure 4.4(b). From the results, STP is better than WTP at providing a consistent proportional delay differentiation between successive classes for link utilization less than 90%. However, WTP becomes slightly superior when link utilization equals or exceeds 90%.

Class Load Distribution

Figure 4.5 shows the ratio of the average delays between successive classes in seven different class load distribution cases. The simulation methodology follows that of the previous section. The SDPs were chosen such that $w_i/w_{i-1} = 2$ for Figure 4.5(a) and $w_i/w_{i-1} = 4$ for Figure 4.5(b). The link utilization is 90% in all cases. The results show that both WTP and STP schedulers are able to provide good delay differentiation, almost independent of the class load distribution. The results in Figure 4.5(a) show that both WTP and STP schedulers are able to provide delay differentiation close to the specified ratio, independent of the class load distribution. The results in Figure 4.5(b), although less precise, show that the performance of WTP and STP are comparable.

Short Timescale Behaviour

The previous two experiments are based on measurements of long-term average delays. In this experiment, the ability of WTP and STP schedulers to provide proportional delay differentiation in short timescales are investigated. The measurements of the ratio of average delays between successive classes are made in consecutive time-intervals of length τ, where τ is the monitoring timescale. The four τ values used in this experiment

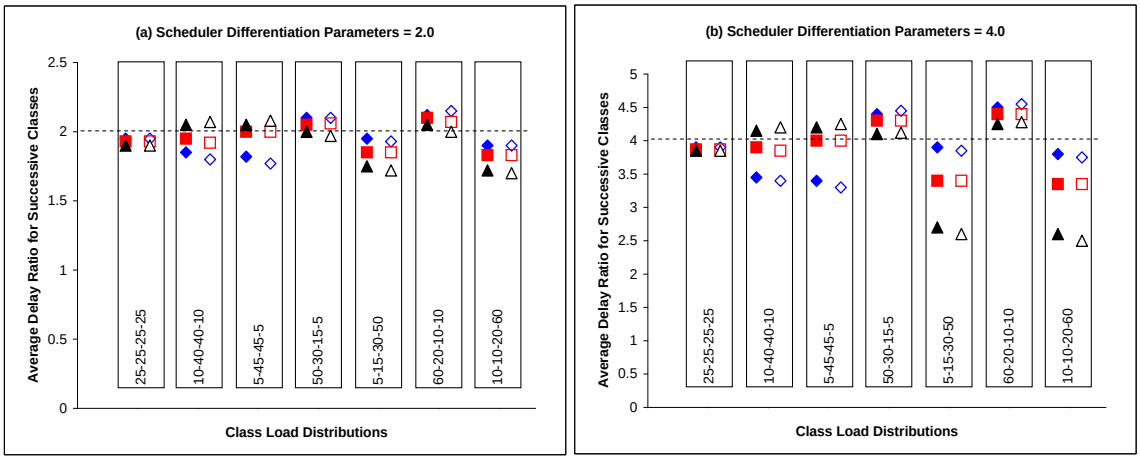

Figure 4.5: The ratio of average-delays between successive classes for WTP and STP with different class load distributions. The symbols in this graph are as in Figure 4.4. The four numbers in each bar denote the fraction of the four classes in the aggregate packet stream, starting from class 1 up to class 4. The utilization is 90% in all cases.

are 100, 1000, 10000 and 100000 μsec. At the end of the simulation run, the ratios of average delays between successive classes for each interval is computed, and averaged over all pairs of classes to get a single measure R. When one or more classes are not active in a certain time interval, the ratios of average delays of the active classes are normalized in the computation of R.

Figure 4.6 shows five percentiles (5%, 25%, 50%, 75% and 95%) of the ratio values obtained from all the different monitoring timescale τ. The SDPs used are chosen such that $w_i/w_{i-1} = 2$ and the link utilization is 90%. From the results, both schedulers are able to provide proportional delay differentiation in the range of 25% and 75%, even for the smallest monitoring timescale. The performance improves as the monitoring time-interval increases.

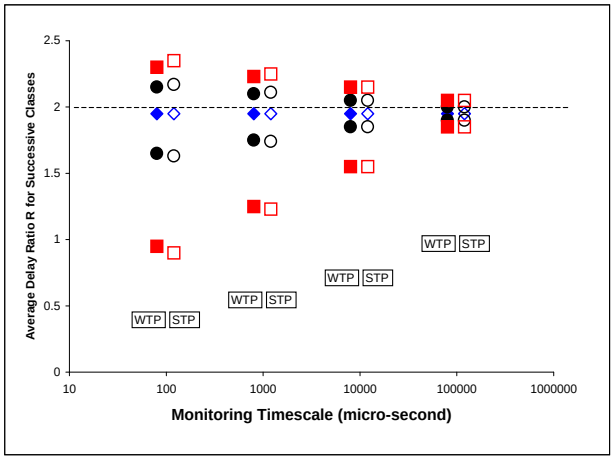

Figure 4.6: Five percentiles of R for four values of the monitoring timescale τ. The diamonds represent the 50% percentiles (median), the circles represent the 25% and 75% percentiles, while the squares represent the 5% and 95% percentiles. The ratio of SDPs is 2.0.

4.5.2 Multiple Nodes

In this experiment, the focus is on the end-to-end performance of the packet flows. The issue here is whether local class-based relative differentiation can lead to consistent end-to-end flow-based relative differentiation. The network topology is the same as Figure 3.4. The number of nodes is varied from 1 to 5. The input traffic at the first node is generated by $P = 4$ sources and traverses all nodes in the network configuration. The traffic characteristics and class load distribution are the same as that of the previous experiments. At each node, cross-traffic that are generated from $C = 4$ sources are included. Their traffic characteristics and class load distribution follows that of the input traffic sources. The SDPs are chosen such that $w_i/w_{i-1} = 2$. The link utilization is 90% with the aggregate cross-traffic load taking up 70% of the total traffic in each link. The bandwidth at each network link is 51.85 Mbps. In order to examine the

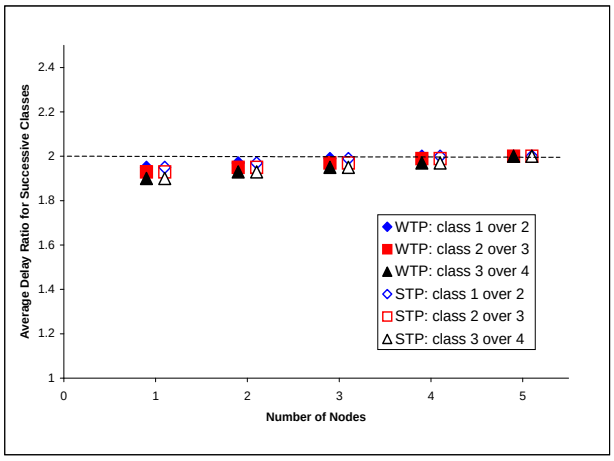

Figure 4.7: R for different number of nodes in a multiple congested nodes network configuration. The scheduler differentiation parameter is 2.0.

effectiveness of the relative delay differentiation, propagation and transmission delays, which are common to all packets, are and focus on queueing delays.

Figure 4.7 shows the ratio of the average delays between successive classes in multiple congested nodes network configuration. Once again the performance of STP and WTP are comparable. It is interesting to observe that as the number of nodes traversed by the input traffic increases, the deviations from the SDP ratio reduces for both STP and WTP. In summary, STP is an excellent approximation of WTP because their performance are comparable under all circumstances.

4.6 Application to QD-PGMQD and QD-PDRD

Referring to equations (3.1), (3.3), and (3.5), it is apparent that the differentiation models for PDD, GMQD, and DRD share a similar form. This similarity in form is also

apparent for the class of Queueing Delay based packet scheduling algorithms designed for PDD, GMQD, and DRD, which are WTP (see equation (4.3)), QD-PGMQD (see equation (3.18)), and QD-PDRD (see equation (3.23)) respectively. Given this similarity, the same method used in deriving STP for WTP can be extended to QD-PGMQD and QD-PDRD algorithms. The simplification of QD-PGQMD and QD-PDRD will be called Scaled QD-PGMQD (SQD-PGMQD) and Scaled QD-PDRD (SQD-PDRD) respectively.

The algorithm for SQD-PGMQD can be defined as follows:

(1) All packets on arrival at the scheduler is timestamped with an arrival tag equal to its arrival time.

(2) A packet on reaching its head-of-queue i at time t has its priority computed as:

$$p_i(t) = \frac{d_i(t)w_i/r_i(t) + D}{t + T} \tag{4.21}$$

where $d_i(t)$ is the waiting-time of the packet at time t, w_i is the SDP that determines the rate with which the priority of the packets of a certain class increases with time, and $r_i(t)$ is the exponential moving average of class i computed using

$$r_k(t) = (1 - e^{-\Delta t/\tau})/\Delta t + e^{-\Delta t/\tau} r_k(t - \Delta t)$$

where Δt is the packet inter-arrival time, and τ is a constant.

T and D are two parameters used to scale the waiting-time of the packet and time the packet reaches its head-of-queue respectively. The condition for SQD-PGMQD to approximate QD-PGMQD is:

$$\min \left| \sum_{i=1}^{P} w_i (1 - \frac{D}{T w_i}) \right| \tag{4.22}$$

(3) The scheduler then chooses the packet with the largest priority tag for service. Ties are arbitrary broken.

(4) Timer of the scheduler resets $t = 0$ at the end of each busy period so that $p_i(t)$ will not gradually tend to zero as t keeps increasing.

Similarly, the algorithm for SQD-PDRD can be defined as follows:

(1) All packets on arrival at the scheduler is timestamped with an arrival tag equal to its arrival time.

(2) A packet on reaching its head-of-queue i at time t has its priority computed as:

$$p_i(t) = \frac{d_i(t)w_i/\phi_i(t) + D}{t + T} \tag{4.23}$$

where $d_i(t)$ is the waiting-time of the packet at time t, and w_i is the SDP that determines the rate with which the priority of the packets of a certain class increases with time.

$$\phi_i(t) = max(r_i(t), \phi_{i+1}(t))$$

where $\phi_{N+1}(t) = 0$.

$r_i(t)$ is the exponential moving average of class i computed using

$$r_k(t) = (1 - e^{-\Delta t/\tau})/\Delta t + e^{-\Delta t/\tau} r_k(t - \Delta t)$$

where Δt is the packet inter-arrival time, and τ is a constant.

T and D are two parameters used to scale the waiting-time of the packet and time the packet reaches its head-of-queue respectively. The condition for SQD-PDRD to approximate QD-PDRD is:

$$\min \left| \sum_{i=1}^{P} w_i (1 - \frac{D}{T w_i}) \right| \tag{4.24}$$

(3) The scheduler then chooses the packet with the largest priority tag for service. Ties are arbitrary broken.

(4) Timer of the scheduler resets $t = 0$ at the end of each busy period so that $p_i(t)$ will not gradually tend to zero as t keeps increasing.

4.7 Related Work

Besides WTP, other more recent schedulers that approximate the PDD model have been proposed. Mean-Delay Proportional (MDP) [47] is an algorithm that uses estimate of average class queueing delay instead of actual waiting time of packets $w_i(t)$ for scheduling. In [13], Dovrolis proposed another WTP-extension, called Hybrid Proportional Delay (HPD). HPD uses a combination of waiting-time of class i packet $w_i(t)$ and average delay of departed class i packets $\bar{d}_i(t)$ to determine the priority of a given packet. However, both algorithms are more computation intensive than WTP.

Another less computation intensive way of achieving proportional delay is to adjust service rate allocations based on the backlog of each class. The Backlog-Proportional Rate (BPR) [43], Proportional Queue Control Mechanism (PQCM) [44], Extended Virtual Clock (Ex-VC) [45] and Dynamic Weighted Fair Queueing (DWFQ) [46] are some algorithms that make use of this method. They differ only in the specific method used to calculate the service rates. However, these methods are generally more susceptible to class load variations [13].

4.8 Conclusion

This chapter presents a novel method for simplifying some algorithms that provide delay differentiation. A new scheduling algorithm called Scale Time Priority (STP) that can efficiently approximate Waiting Time Priority (WTP) is proposed. By deriving the workload that must be transmitted before an arbitrary packet for WTP and STP schedulers, the less computationally intense STP is shown to be a good approximation to WTP. Specifically, STP is able to provide near proportional delay at a complexity of $O(1)$, which is lower than the $O(N)$ complexity of WTP, where N is the number of service classes in the system. This same method was subsequently applied to simplify Queueing Delay based Packetized Generalized Minimum Queueing Delay (QD-PGMQD) and Queueing Delay based Packetized Delay Rate Differentiation (QD-PDRD).

Chapter 5

A Control-Theoretical Approach for Achieving Fair Bandwidth Allocations in Core-Stateless Networks

This chapter describes the second portion of the Delay-Rate Differentiated Services (DRDS) framework, which is to provide flow isolation within a class of aggregated best-effort traffic flows so that congestion unresponsive flows, like User Datagram Protocol (UDP), do not squeeze out congestion responsive flows, like Transmission Control Protocol (TCP).

5.1 Background

Until recently, fair bandwidth allocations were best achieved using per-flow queueing mechanisms like Weighted Fair Queueing [57] and its many other variants [5], [6], [7], [75]. In Weighted Fair Queueing, each flow has its own First-In-First-Out (FIFO) queue. Packets are transmitted using an approximate bit-by-bit round-robin discipline. These

proposed mechanisms are usually based on a *stateful network* architecture, i.e. a network in which every router maintains per flow state. In addition, most of these mechanisms require per-flow queueing. As there can be a large number of flows in the Internet, the complexity required to implement these mechanisms severely limits their deployment over high speed backbone core routers. Furthermore, there is the need for complex algorithms to set and preserve the consistency of states across the network, making robustness a lot harder to achieve.

In order to reduce this complexity of doing per-flow queueing and maintaining per-flow state information new algorithms have been introduced. Stoica et al [37] was the first to propose a fair queueing scheduling algorithm that does not require per-flow state information, called Core-Stateless Fair Queueing (CSFQ). CSFQ is based on the Stateless Core (SCORE) architecture [12], which is similar to the Differentiated Services (Diffserv) architecture [10].

In CSFQ, packets are labelled with their flow arrival rate at the edge, and they are dropped probabilistically when their arrival rate exceeds the fair share estimated by the core routers. Through extensive simulations, CSFQ was found to approach the fairness of Deficit Round Robin (DRR) [75] and offer significant improvements over FIFO and Random Early Drop (RED) [76]. Besides the CSFQ framework, the other major framework for achieving fair bandwidth sharing in a SCORE/DPS network is the Rainbow Fair Queueing (RFQ) algorithm [38]. In RFQ, packets of each flow are divided into a set of "color" layers at the edge and they are dropped if their "color" label is greater than the color threshold value determined at the core routers. Unfortunately,

the methods used by CSFQ and RFQ in their respective fair share and color threshold estimation are rather ad-hoc in nature and require additional logic for saturated traffic conditions [37], [38]. In particular, CSFQ has been shown to perform badly under bursty cross-traffic [38], [39].

The main contribution of this chapter is the use of a control-theoretical approach for fast and robust computation of the fair share or color threshold value [50], [51]. The use of a control-theoretical approach is appealing as it can leverage on the concepts found in control theory to produce stable and robust systems. In particular, it has been found that with proper modelling, the resultant system can exhibit stable, robust, good transient and steady state behavior. For the purpose of illustration, the Linear Quadratic (LQ) control method from optimal control theory is applied to modifications of CSFQ and RFQ. Simulations are then used to show the superior performance of this approach as compared to the original CSFQ scheme. Note that besides the LQ control method, other control algorithms can also be used. However, the focus here is not on evaluating the comparative performance of various control algorithms.

This chapter is structured as follows: In Section 5.2, the process on achieving fair bandwidth sharing in the SCORE/DPS architecture is described. This is followed by details on how CSFQ and RFQ achieves fair bandwidth sharing. In Section 5.3, the control-theoretical model of the proposed system and the key implementation issues involved are discussed. In Section 5.4, the system is evaluated through simulations. Some related works are discussed in Section 5.5 before the conclusion in Section 5.6.

Notation	Comments
C	link capacity
N	number of flows
r_i	rate of flow i
w_i	the weight assigned to flow i
r_{fair}	fair share rate
Δt	packet inter-arrival time
α	estimate of fair share rate
$\hat{F}$	estimate of aggregate traffic acceptance rate
CT	color threshold value

Table 5.1: Notations used in Section 5.2.

5.2 Stateless Core/ Dynamic Packet State Framework for Providing Flow Isolation

5.2.1 Objective

The SCORE network deals with traffic aggregates at the core routers and does not need to perform per-flow state management. It makes use of DPS to achieve a functional approximation of a stateful network. The primary objective of SCORE/ DPS fair queueing algorithms is therefore to provide flow isolation without the need to provide or maintain per flow state information. In other words, the objective is to achieve max-min fairness [77] with minimum implementation complexity. To achieve this objective, these algorithms make use of the following idea: Consider a link with capacity C serving N number of flows with rates given as $r_1, r_2, ..., r_N$ (see Table 5.1). Assume weights w_i are assigned to different flows, such that a flow assigned a weight of 2 will get twice as much bandwidth compared to a flow with weight 1. Max-min fairness is then achieved

when the fair share r_{fair} is the unique solution to

$$C = \sum_{i=1}^{N} w_i \ min(\frac{r_i}{w_i}, r_{fair}) \tag{5.1}$$

Note that when congestion occurs, weighted flow rates r_i/w_i above r_{fair} will be constrained to r_{fair}, while weighted flow rates r_i/w_i equal or below r_{fair} remain unchanged. On the other hand, when $C \geq \sum_{i=1}^{N} r_i$, all flows can pass through unconstrained and r_{fair} becomes equal to the highest weighted flow rate.

5.2.2 Core-Stateless Fair Queueing Framework

To facilitate the following discussion, consider how CSFQ achieves the above objective. CSFQ does it through the following four steps:

(1) When a flow arrives at the edge of the network, its rate r is estimated by exponential averaging calculated as:

$$r_{new} = (1 - e^{-\Delta t/\tau})/\Delta t + e^{-\Delta t/\tau} r_{old} \tag{5.2}$$

where Δt is the packet inter-arrival time, and τ is a constant[1].

(2) The edge router then labels each packet of the flow with a state value that is proportional to this estimated rate. Due to the need to represent a large range of flow rates with a limited number of state values, a non-linear representation is used to limit the error of representation to a fixed range. For CSFQ, a simple floating point representation consisting of mantissa and exponential component is used [12].

(3) Inside the network, packets from different flows are interleaved together. Core

[1] A good discussion on what value to set for τ can be found in [12].

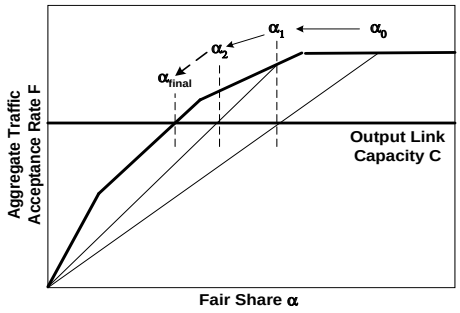

Figure 5.1: Basic framework on how CSFQ estimates the fair share, α

routers use FIFO queueing and do not keep per-flow state. At the network core, each router estimates a fair share rate α by approximating the aggregate traffic acceptance rate F of the router using a linear function that intersects the origin with a slope of $\hat{F}/\alpha_{old}$, yielding

$$\alpha_{new} = \alpha_{old}\frac{C}{\hat{F}} \tag{5.3}$$

where $\hat{F}$ is the estimated aggregate traffic acceptance rate computed using exponential averaging.

(4) When congestion occurs, packets of every flow i in the system are dropped with probability

$$Prob = max(0, 1 - \frac{\alpha}{r_i/w_i}) \tag{5.4}$$

where r_i and w_i denote respectively the rate and weight of flow i found in the header of the packet. Finally, packets are relabelled using the minimum of the incoming flow rate and the router's estimated fair share rate α.

Steps (3) and (4) illustrates how max-min fairness is achieved using the CSFQ algorithm. When F is larger than C, α will be reduced due to the $C/\hat{F}$ ratio (see

equation (5.3)), leading to a higher packet dropping probability for flows with r/w larger than α (see equation (5.4)). On the other hand, when F is smaller than C, α will be increased, thereby reducing the packet dropping probability. Therefore, the process of adjusting α so that F converges to C leads to max-min fairness because $F = \sum_{i=1}^{N} w_i \ min(\frac{r_i}{w_i}, \alpha)$ (see equation (5.1)). Figure 5.1 illustrates how α will eventually converge to $\alpha_{final} = r_{fair}$ for the case when F is larger than C.

The method used in CSFQ for estimating fair share α is simple but does not function well when the aggregate incoming traffic is bursty [38], [39]. In fact, in order to minimize the negative effects of buffer overflow, Stoica included a simple heuristic whereby each time the buffer overflows, α is decreased by a small fixed percentage, taken to be 1% in his simulations [37]. In addition, to avoid over-correction, it is ensured that during consecutive updates α does not decrease by more than 25%. The CSFQ algorithm is therefore, unable to quickly and robustly compute α for very bursty traffic, which is crucial for achieving max-min fairness.

5.2.3 Rainbow Fair Queueing Framework

Besides the CSFQ framework, the other major framework is the Rainbow Fair Queueing (RFQ) algorithm [38]. Specifically, RFQ achieves max-min fairness through the following three steps:

(1) As in CSFQ, when a flow arrives at the edge of the network, its rate is estimated by exponential averaging.

(2) The edge router then divides the flow into many thin layers where each layer is as-

signed a state value. The number of layers is proportional to the flow rate. A non-linear rate representation which is similar, but not identical, to CSFQ is used here [38].

(3) Inside the network, packets from different flows are interleaved together to form a single aggregated flow with many different layers. Core routers use FIFO queueing and maintain a color threshold CT value, whereby packets with state values higher than CT are discarded. At the first occurrence of congestion, traffic discarding starts with packets having the highest state value. In other words, CT starts with the highest flow rate in the router. If congestion persists, CT is decreased and more packets are discarded until there is no more congestion. In this way, core routers discard the layers in a top down manner and because the layering is done proportionally to the flow rate, the algorithm approximates that of a fair queueing system. On the other hand, when the link is under-utilized, CT is increased to allow more traffic to pass through.

In essence, under the RFQ scheme, CT is adjusted to drive F to C (see Section 5.2.2) in order to achieve max-min fairness.

5.2.4 Discussion

A comparison between the CSFQ and RFQ implementation frameworks is presented in Table 5.2. As mentioned in [38], the layering method in RFQ has several advantages over CSFQ. Firstly, packet re-labelling inside the core is not required because the packet header keeps its label value throughout the lifetime of the packet inside the core. Secondly, input rate estimation at the core router, which is required in CSFQ algorithm [37] is not required for RFQ, further reducing core router complexity. Finally, preference for

	CSFQ Framework	**RFQ Framework**
Rate estimation at ingress	Exponential averaging	Exponential averaging
Label/state information	Explicit rate of the flow	Color layers that the packets belong to
Packet dropping algorithm	Controlled by fair share	Controlled by color threshold
Packet re-labelling	Required	Not required
Input rate estimation at core	Required	Not required
Ability to express preference for certain packets	No	Yes

Table 5.2: A comparison of the implementation frameworks of CSFQ and RFQ algorithms

more important packets can be expressed using the RFQ framework.

The overall architecture of SCORE/DPS fair queueing algorithm is shown in Figure 5.2. The portion that needs to be highlighted is the packet dropping algorithm at the core router, which corresponds to steps (3) and (4) of the CSFQ algorithm described in Section 5.2.2 and step (3) of the RFQ algorithm described in Section 5.2.3. The packet dropping algorithm describes the process for adjusting α or CT in order to achieve max-min fairness. As discussed earlier, the methods used by CSFQ and RFQ are rather ad-hoc in nature and require additional logic for saturated traffic conditions. In particular, CSFQ has been shown to perform badly under bursty cross-traffic [38], [39]. More importantly, note that the key α or CT computation process can be based on a closed-loop rate-based scheme if α or CT is adjusted based on the feedback of past F values and buffer occupancy values of the system. Control-theoretical approaches can, therefore, be readily applied to this closed-loop rate-based system. This forms the basis of the work which will be described in the rest of this chapter.

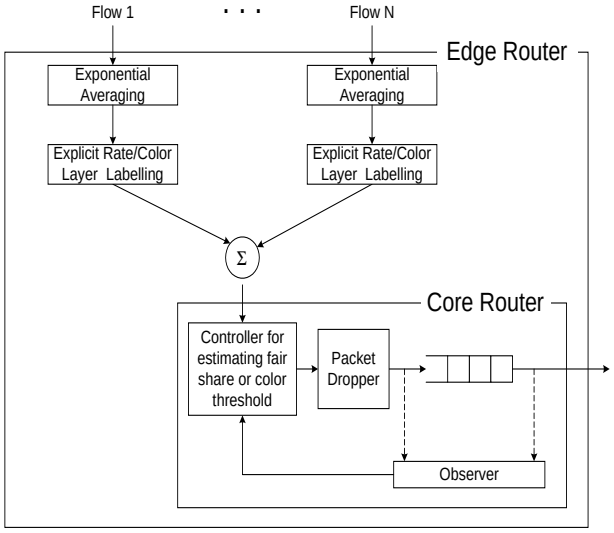

Figure 5.2: Overall architecture of SCORE/DPS fair queueing algorithms.

5.3 Control Theoretical Approach

In this section, the discrete time control model of the proposed system is described, before the Linear Quadratic control method, from optimal control theory, is applied to it. Implementation issues will also be discussed.

5.3.1 Closed-Loop Dynamics

Assume a system whereby the traffic and buffer occupancy are sampled and updated at periodic intervals T. Let $F(n)$ denote the aggregate traffic accepted during the n^{th} timeslot. In addition, let $Q(n)$ denote the buffer occupancy of the FIFO queue at the beginning of the n^{th} timeslot. The closed loop dynamics of the proposed system is

therefore, given by

$$F(n+1) \quad = Sat_0^{C_{in}} \quad \left\{ F(n) - \sum_{j=0}^{J} \lambda_j (Q(n-j) - Q^0) \right. \\ \left. - \sum_{k=0}^{K} \mu_k F(n-k) \right\} \tag{5.5}$$

$$Q(n+1) = Sat_0^B \{Q(n) + F(n) - \psi\} \tag{5.6}$$

where

$$Sat_b^a(x) = \begin{cases} b & \text{if} \quad x < b \\ a & \text{if} \quad x > a \\ x & \text{otherwise} \end{cases} \tag{5.7}$$

The other parameters involved in the computations above are:

(1) Update/sampling period T: this corresponds to the duration of a single timeslot, where computations of F and Q are done periodically.

(2) Buffer threshold Q^0: this is the buffer occupancy value that the controller tries to achieve and corresponds to the desired steady state buffer occupancy value. In general, Q^0 is chosen to satisfy two objectives, it must be small enough to minimize queueing delay but large enough to ensure full utilization of the link capacity.

(3) C_{in}: this is the maximum aggregate traffic that can be accepted in a single timeslot.

(4) Feedback control gains λ_j and μ_k: these values will be determined later using the LQ algorithm. It will be shown from steady state analysis that λ_j and μ_k must satisfy the following requirements:

$$\sum_{j=0}^{J} \lambda_j \neq 0, \qquad \sum_{k=0}^{K} \mu_k = 0$$

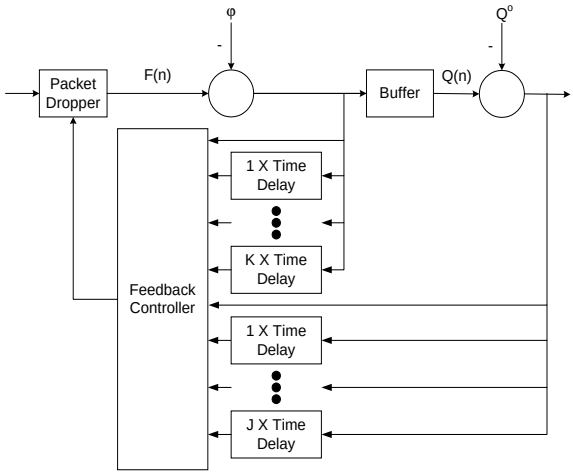

Figure 5.3: Block diagram of the proposed control system.

(5) J and K are non-negative integers. It will be shown later from stability analysis that $J = 1$ and $K = D$, where $D \times T$ is the maximum feedback delay of the system.

(6) Buffer size B: this is used to impose bounds on the buffer size of the switch.

(7) Traffic service rate ψ: this denotes the aggregate traffic serviced during a single timeslot. ψ is assumed to be a constant equal to the output link capacity multiplied by T. Note that if high priority traffic that has to be serviced first is considered, then ψ will be a variable time function. However, since ψ can also be considered as an input to the system (as indicated in equation (5.6)), a stable closed loop system will imply that this system is able to track any variation to the value of ψ.

The block diagram of the system is illustrated in Figure 5.3.

5.3.2 Steady State Analysis

Next, establish the steady state conditions of the system. Let F_s and Q_s be the steady state values corresponding to equations (5.5) and (5.6) under the assumption that the input traffic is constant. Therefore,

$$F_s = Sat_0^{C_{in}} \left\{ F_s - \sum_{j=0}^{J} \lambda_j (Q_s - Q^0) - \sum_{k=0}^{K} \mu_k F_s \right\} \tag{5.8}$$

$$Q_s = Sat_0^{B} \{ Q_s + F_s - \psi \} \tag{5.9}$$

resulting in

$$F_s = \psi \quad \text{for} \quad 0 \leq Q_s \leq B \quad and \quad 0 \leq F_s \leq C_{in} \tag{5.10}$$

$$Q_s = Q^0 - \frac{\sum_{k=0}^{K} \mu_k}{\sum_{j=0}^{J} \lambda_j} F_s \quad \text{for} \quad 0 \leq Q_s \leq B \tag{5.11}$$

To ensure that $Q_s = Q^0$ for non-zero F_s, the constraint $\sum_{k=0}^{K} \mu_k = 0$ is included, which in other words mean $\mu_K = -\sum_{k=0}^{K-1} \mu_k$. In addition, $\sum_{j=0}^{J} \lambda_j \neq 0$ is required to ensure the stability of the system.

5.3.3 Stability

With the closed-loop dynamics and steady state conditions in place, the asymptotic stability properties of the closed loop system is analyzed next by removing the saturation non-linearity of equations (5.5) and (5.6). This results in

$$F(n+1) = F(n) - \sum_{j=0}^{J} \lambda_j (Q(n-j) - Q^0) - \sum_{k=0}^{K} \mu_k F(n-k) \tag{5.12}$$

$$Q(n+1) = Q(n) + F(n) - \psi \tag{5.13}$$

One of the earliest analysis of the above closed-loop system was given in [78], where a long proof was used to establish that asymptotic stability can be achieved if $J = 1$ and $K = D$ are chosen. D is a non-negative integer, where $D \times T$ corresponds to the maximum delay lag that a traffic source takes to react to feedback given by the switch, i.e. maximum feedback delay. Viewed in another way, $F(n), F(n-1), \cdots, F(n-D)$ are all the previously sampled rate information that will influence future buffer occupancy behavior $Q(m)$ for $m > n$.

Converting equations (5.12) and (5.13) into the state-space vector form,

$$X(n+1) = \begin{pmatrix} 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ -\lambda_0 & -\lambda_1 & 1-\mu_0 & -\mu_1 & \cdots & -\mu_D \\ & & 1 & 0 & \cdots & 0 \\ & & & \ddots & \ddots & \vdots \\ & & & & 1 & 0 \end{pmatrix} X(n) + \begin{pmatrix} -1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \psi \tag{5.14}$$

where

$$X(n) = [(Q(n) - Q^0), Q(n-1) - Q^0), F(n), \cdots, F(n-D)]'$$

with the prime symbol ($'$) denoting the transpose operator.

It was also shown in [78] that $(D + 2)$ of the $(D + 3)$ poles of this closed-loop system can be placed at will within the unit circle by an appropriate choice of the gains $\lambda_0, \lambda_1, \mu_0, \cdots, \mu_D$, with the remaining pole placed at 0. Note that all poles must lie within the unit circle for stability reasons.

5.3.4 Gain Selection

An alternative approach that provides an easier way of computing the feedback controller gains and to balance between the speed of convergence and magnitude of the state variables is based on the Linear Quadratic (LQ) control method [79], [80].

Given a system with state space model:

$$Z(n+1) = AZ(n) + BV(n) \tag{5.15}$$

the LQ design problem involves solving the control function $V(n)$ that minimizes the cost function:

$$L = \sum_{n=0}^{\infty} \Big[Z^{'}(n) W_1 Z(n) + V^{'}(n) W_2 V(n) \Big] \tag{5.16}$$

where W_1 and W_2 are the design parameters of the LQ control problem that allow different emphasis to be placed on the states and inputs. The first term on the right is the cost associated with state deviation, while the second term is the cost associated

with inputs to the system. Note that both W_1 and W_2 must be symmetric and positive definite.

The optimal solution to the minimization problem, obtained using dynamic programming [80] is

$$V(n) = -HZ(n)$$

where

$$H = W_2 + (B^{'}SB)^{-1}B^{'}SA$$

and S is the solution of the following matrix equation known as the *Riccati equation*:

$$A^{'}SA - S - A^{'}SB(W_2 + B^{'}SB)^{-1}B^{'}SA + W_1 = 0$$

The vector of closed-loop poles using the LQ algorithm are the eigenvalues of the matrix $A - BH$. Note that the resulting system is stable even if the actual feedback delay deviates slightly from D.

Next, convert the dynamic equations (5.12) and (5.13) into a form suitable for the LQ algorithm. Taking into account $\mu_K = -\sum_{k=0}^{K-1} \mu_k$, where $K = D$, equation (5.12) can be converted into:

$$F(n+1) = F(n) - \sum_{j=0}^{1} \lambda_j (Q(n-j) - Q^0) - \sum_{k=0}^{D-1} \mu_k (F(n-k) - F(n-D))$$

$$\begin{aligned} F(n+1) - F(n+1-D) &= [F(n) - F(n-D)] - \\ &\quad [F(n+1-D) - F(n-D)] - \end{aligned}$$

$$\sum_{j=0}^{1} \lambda_j (Q(n-j) - Q^0) - \sum_{k=0}^{D-1} \mu_k (F(n-k) - F(n-D)) \tag{5.17}$$

Similarly, equation (5.13) can be converted into

$$\begin{aligned} Q(n+1) - Q^0 &= Q(n) - Q^0 + F(n) - \psi + \\ &\quad Q(n) - Q^0 - [Q(n-1) - Q^0 + F(n-1) - \psi] \\ &= 2[Q(n) - Q^0] - [Q(n-1) - Q^0] + \\ &\quad [F(n) - F(n-D)] - [F(n-1) - F(n-D)] \end{aligned} \tag{5.18}$$

The state-space vector representation of equations (5.17) and (5.18) is

$$Y(n+1) = \begin{pmatrix} 2 & -1 & 1 & -1 & & 0 \\ 1 & 0 & 0 & 0 & & 0 \\ 0 & 0 & 1 & 0 & & -1 \\ & & 1 & & & -1 \\ & & & \ddots & & \vdots \\ & & & & 1 & -1 \end{pmatrix} Y(n) + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} U(n) \tag{5.19}$$

where

$$\begin{aligned} Y(n) &= [(Q(n) - Q^0), (Q(n-1) - Q^0), F(n) - F(n-D), \cdots, \\ &\quad F(n-K) - F(n-D)]' \end{aligned}$$

$$\begin{aligned} U(n) &= -\sum_{j=0}^{1} \lambda_j [Q(n-j) - Q^0] - \sum_{k=0}^{D-1} \mu_k [F(n-k) - F(n-D)] \\ &= -GY(n) \end{aligned}$$

$$G = (\lambda_0, \lambda_1, \mu_0, \mu_1, \cdots, \mu_{D-1})$$

For the proposed system, state information is sampled periodically at the core router. The packet dropper, that controls the incoming traffic at the core router, is therefore able to react quickly based on state information obtained one timeslot away. Making the assumption that negligible delay uncertainty is introduced by the sampler, $D = 1$ can be chosen without the fear of feedback delay deviation affecting system stability.

This greatly simplifies the system, resulting in

$$Z(n+1) = \begin{pmatrix} 2 & -1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} Z(n) + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} V(n) \tag{5.20}$$

where

$$Z(n) = [(Q(n) - Q^0), (Q(n-1) - Q^0), F(n) - F(n-1)]'$$

$$\begin{aligned} V(n) &= -\sum_{j=0}^{1} \lambda_j [Q(n-j) - Q^0] - \mu_0 [F(n) - F(n-1)] \\ &= -HZ(n) \end{aligned}$$

$$H = (\lambda_0, \lambda_1, \mu_0)$$

The LQ control method can now be readily applied to the above form.

5.3.5 Implementation Issues

The design parameters of the LQ control problem are the weighting matrices W_1 and W_2 in the cost function L. W_1 is a (3×3) matrix, while W_2 is a scalar because $V(n)$ is a scalar in the proposed system. For high-speed implementations, a look-up table can be used to store the pre-computed values of feedback gains as a function of pre-determined weighting matrices W_1 and W_2. To construct the look-up table, choose the W_1 values and vary W_2 to generate different gains. Good examples of W_1 values are diagonal matrices with zero elements at all positions except for:

(a) $W_1(1,1) = 1$, which emphasizes on reducing the deviation $Q(n) - Q^0$.

(b) $W_1(1,1) = 1$ and $W_1(2,2) = 1$, which emphasizes on reducing the deviations $Q(n) - Q^0$ and $Q(n-1) - Q^0$.

(c) $W_1(1,1) = 1$ and $W_1(3,3) = 1$, which emphasizes on reducing the deviations $Q(n) - Q^0$ and $F(n) - F(n-1)$.

(d) $W_1(1,1) = 1$, $W_1(2,2) = 1$ and $W_1(3,3) = 1$, which emphasizes on reducing the deviations $Q(n) - Q^0$, $Q(n-1) - Q^0$ and $F(n) - F(n)$.

Note that emphasizing on reducing the deviation $Q(n) - Q^0$ is important because it reduces buffer overflow. In addition, an appropriate Q^0, like 10% to 20% of total buffer size, can help ensure good link utilization.

The values of W_2 are selected based on the output response that the control system produces. Note that making W_2 smaller decreases the penalty on $V(n)$, resulting in an

increase in the magnitude of $V(n)$. Since $V(n) = F(n+1) - F(n)$, this means that F will change faster, resulting in faster convergence. In general, faster convergence leads to a greater oscillatory transient response. Hence, the desired output response of the control system is limited in range.

Another issue is the slow response of the control algorithm that occurs when there is a lack of incoming traffic for an extended period of time. In order to address this issue, the concept of virtual queue (VQ) is included in the implementation. Specifically, VQ works as follows:

(a) When the buffer occupancy is greater than one packet, VQ is equal to the actual queue length.

(b) When the buffer occupancy is less than or equal to one packet, VQ is computed by subtracting any excess bandwidth from the previous VQ value.

Note that a threshold of one packet is used in step (a) because the sampled queue value will not be zero if there is a packet currently being served, even when there is little incoming traffic. The VQ concept results in a larger error term ($VQ(n) - Q^0$) when there is a lack of incoming traffic for an extended period of time, thereby addressing the issue of slow response of the control algorithm.

5.3.6 Control-Theoretical Approach to CSFQ and RFQ

The steps involved when the proposed control-theoretical approach is applied to the packet dropping algorithm of CSFQ (steps (3) and (4) described in the CSFQ framework of Section 5.2.2) and RFQ (step (3) described in the RFQ framework of Section 5.2.3) are

now described. The modification of CSFQ and RFQ will be called Control-theoretical Approach to CSFQ (CA-CSFQ) and Control-theoretical Approach to RFQ (CA-RFQ) respectively.

The packet dropping algorithm of CA-CSFQ involves the following steps:

(1) Choose W_1 and W_2.

(2) Obtain the pre-computed feedback gains $\lambda_0, \lambda_1, \mu_0$ from the look-up table.

(3) The fair share α is computed using:

$$\alpha_{new} = \alpha_{old} - \sum_{j=0}^{1} \lambda_j [VQ(n-j) - Q^0] - \mu_0 [F(n) - F(n-1)] \tag{5.21}$$

Equation (5.21) is modified from the last row of equation (5.20). Note that the aggregate traffic F consists of traffic flows with rates less than or equal to α. Therefore, F is a continuous, non-decreasing, concave, and piecewise-linear function of α. Hence, α can be adjusted based on the feedback values of F and VQ.

(4) When congestion occurs, packets of every flow i in the system are dropped with probability

$$Prob = \max(0, 1 - \frac{\alpha}{r_i/w_i}) \tag{5.22}$$

where r_i and w_i denote respectively the rate and weight of flow i, which are contained in the header of the packet.

Similarly, the packet dropping algorithm of CA-RFQ will involve the following steps:

(1) Choose W_1 and W_2.

(2) Obtain the pre-computed feedback gains $\lambda_0, \lambda_1, \mu_0$ from the look-up table.

(3) The color threshold CT is computed using:

$$CT_{new} = CT_{old} - \sum_{j=0}^{1} \lambda_j [VQ(n-j) - Q^0] - \mu_0 [F(n) - F(n-1)] \tag{5.23}$$

which is similar to the CA-CSFQ algorithm.

(4) When congestion occurs, packets belonging to layers higher than the $\lceil CT \rceil$ will be dropped, where $\lceil CT \rceil$ is the round-up integer value of CT.

Note that the an integer $\lceil CT \rceil$ value is required for RFQ because packets are discarded in a layered fashion, unlike in CSFQ where packets are dropped probabilistically.

5.4 Simulations

In the following section, the proposed approach is illustrated with examples and their performance evaluated using simulations. Specifically, the performance of DRR, RED and CSFQ are compared with CA-CSFQ and CA-RFQ.

Deficit Round Robin (DRR) [75] is an efficient implementation of WFQ that has a complexity of O(1). In DRR, queues are served in a weighted round robin fashion and a packet is dropped from the longest queue when the buffer is full. DRR requires per-flow queueing and is used as the benchmark for fair bandwidth sharing.

Random Early Drop (RED) is one of the most well-known algorithm for buffer management. In RED, all flows shared a single First-In-First-Out (FIFO) queue. The RED algorithm consist of two parts: (a) in the first part, RED estimates the time-average queue size using exponential moving average. (b) in the second part, RED drops packets probabilistically based on the time-averaged queue size. The probability of drop

increases linearly from zero to max_p (maximum dropping probability) as the average queue size grows from min_{th} (minimum threshold) to max_{th} (maximum threshold).

In the simulations, the output link capacity C is set at 10 Mbps and the link buffer size is set at 64 Kbytes. The packet size is fixed at 1 Kbytes. For CA-RFQ, the non-linear encoding algorithm proposed by Cao in [38] with parameters $a = 3$, $b = 32$ and $P = 65$ Mbps is used. For CA-CSFQ and CA-RFQ, Q^0 is set at 20% of link buffer size B, ψ at C and T at 1 msec. Detailed descriptions of other simulation parameters of CSFQ can be found in [37].

The cost function chosen for CA-CSFQ and CA-RFQ is

$$L = \sum_{n=0}^{\infty} \left[(Q(n) - Q^0)^2 + V^2(n) \right]$$

This simple cost function emphasizes only on the input $V(n)$ and on reducing the deviation $Q(n) - Q^0$. Note that with $Q^0 = 0.2B$, an emphasis on $Q(n) - Q^0$ will help ensure that there is no buffer overflow and full link utilization.

Solving the cost function above yields a feedback vector H given by

$$H = [1.4731, -1.0891, 1.0891].$$

5.4.1 Single Link

The first network topology considered is a simple multiple input, single output case. In the first experiment, 32 UDP flows sharing a single bottleneck link with each flow having equal weight are used. Each flow sends at $10i/32$ Mbps, where $i \in (1...32)$ is the flow number. During the 10 seconds simulation, each UDP flow has an infinite amount of data to transmit and hence the link is severely congested. Under max-min fairness,

each flow should achieve an average throughput of 0.3125 Mbps. The results in Figure 5.4(a) show that while DRR gives almost perfect bandwidth sharing among contending flows, RED cannot ensure fair sharing. In comparison, CSFQ, CA-CSFQ and CA-RFQ all have comparable performance that are slightly inferior to DRR but much better than RED.

In the second experiment, the performance of a TCP flow competing against a set of $(N-1)$ non-reactive UDP flows transmitting at twice their fair share is evaluated. Figure 5.4(b) shows the normalized throughput achieved by the TCP flow over the 10 seconds simulation time. The results show that the flow isolation abilities of both CA-CSFQ and CA-RFQ are comparable: they perform slightly better than CSFQ and significantly out-perform RED. For DRR (as observed in [37], [76]), the TCP flow's flow is significantly affected by the limited buffer share when there are more than 22 flows.

5.4.2 Multiple Links

Next, the performance of CA-CSFQ and CA-RFQ over multiple links are evaluated. The network topology is the same as Figure 3.4. The number of nodes varies from 1 to 5. The output link capacity remains unchanged at 10 Mbps with propagation delay 1 msec. At each of the nodes, five cross-traffic flows carrying UDP traffic at an average of 4 Mbps each are connected. In the first experiment, the reference flow is a UDP flow transmitting at 4 Mbps. Figure 5.5(a) shows the normalized throughput. DRR has the best performance, while CSFQ, CA-CSFQ and CA-RFQ have comparable performance which are slightly inferior to DRR. RED has the worst performance.

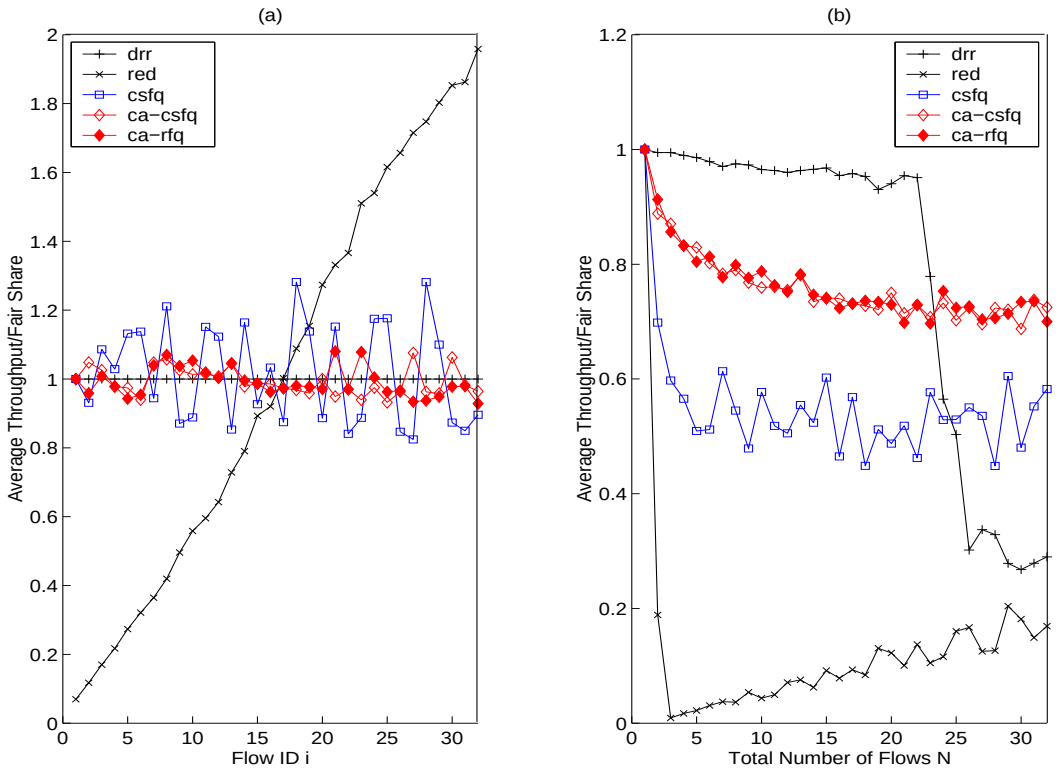

Figure 5.4: The normalized throughput achieved by: (a) each of the 32 UDP flows sharing a bottleneck link where flow i sends at i times its fair share (0.3125 Mbps), (b) a TCP flow competing against $(N-1)$ UDP flows, each sending at twice their fair share.

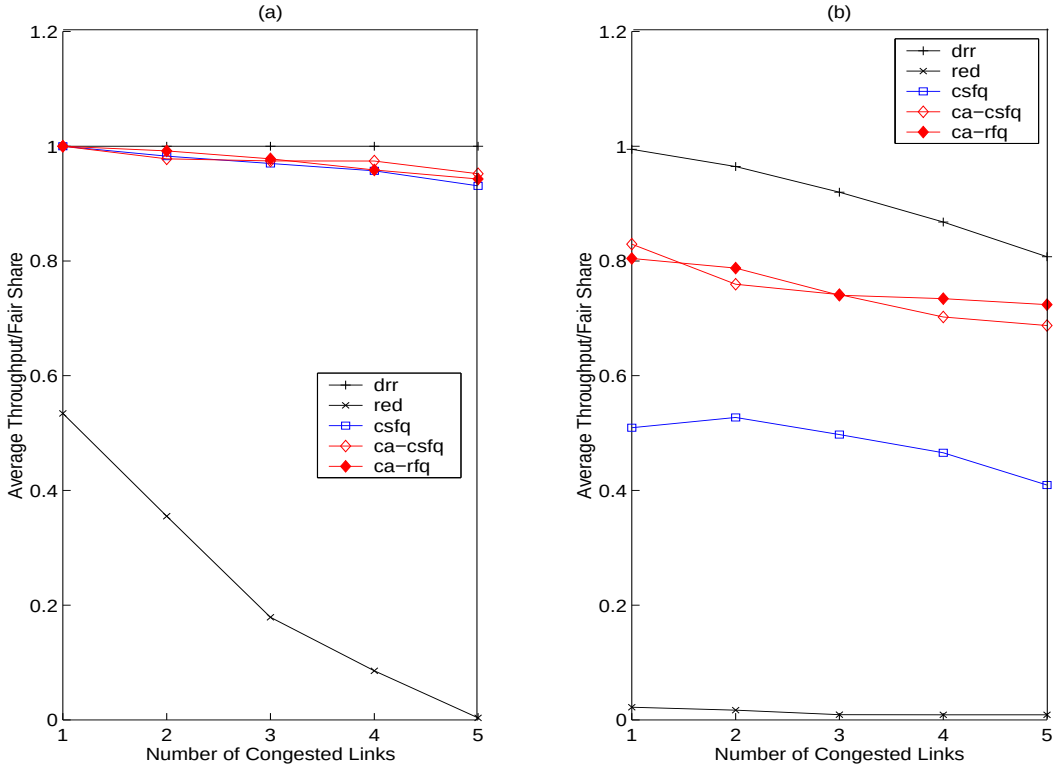

Figure 5.5: (a) Normalized throughput of a UDP flow as a function of the number of congested links. Cross traffic are UDP sources sending at twice the fair share. (b) The same plot as (a) but with the UDP flow being replaced by a TCP flow.

In the second experiment, the reference flow was changed to a TCP flow. Figure 5.5(b) shows the normalized bandwidth share it receives. The results show that the performance of CA-CSFQ and CA-RFQ are comparable, but slightly better than CSFQ. On the other hand, RED fails to protect the TCP flow.

5.4.3 Bursty Cross Traffic

In the last set of experiments, the effects of bursty cross-traffic sources is evaluated. The simulations use the same topology as Figure 3.4, but the UDP sources that form the

cross traffic are now replaced with ON/OFF sources. The burst (ON) and idle (OFF) time periods are both exponentially distributed with the same average chosen between 20 msec and 0.5 sec. The average intensity of cross traffic remains unchanged from the previous set of simulations, that is, the ON/OFF sources send at 4 Mbps during the ON period. In addition, only the results for SCORE/DPS algorithms CSFQ, CA-CSFQ and CA-RFQ will be compared.

Note that the ability of the algorithms to quickly respond to a burst or drop in traffic conditions actually shows how responsive the algorithms are to transient load variations. By measuring the performance of each algorithm for different ON/OFF periods, the control responsiveness of each algorithm over a range of bursty conditions can be established.

The results in Figure 5.6 showed that as the ON/OFF time periods reach the critical 100 msec, the performance of CSFQ becomes seriously affected. The reason is that the fair share estimation algorithm in CSFQ is unable to compute the correct fair share value quickly. In comparison, both CA-CSFQ and CA-RFQ are still able to achieve a reasonable normalized throughput.

5.5 Related Work

There are several rate-based flow control systems proposed in the literature of network Quality-of-Service (QoS) management, with the better known systems being those proposed for the Available Bit Rate (ABR) service category in Asynchronous Transfer Mode (ATM) networks [81]. The ABR congestion control adopted a closed-loop rate-based

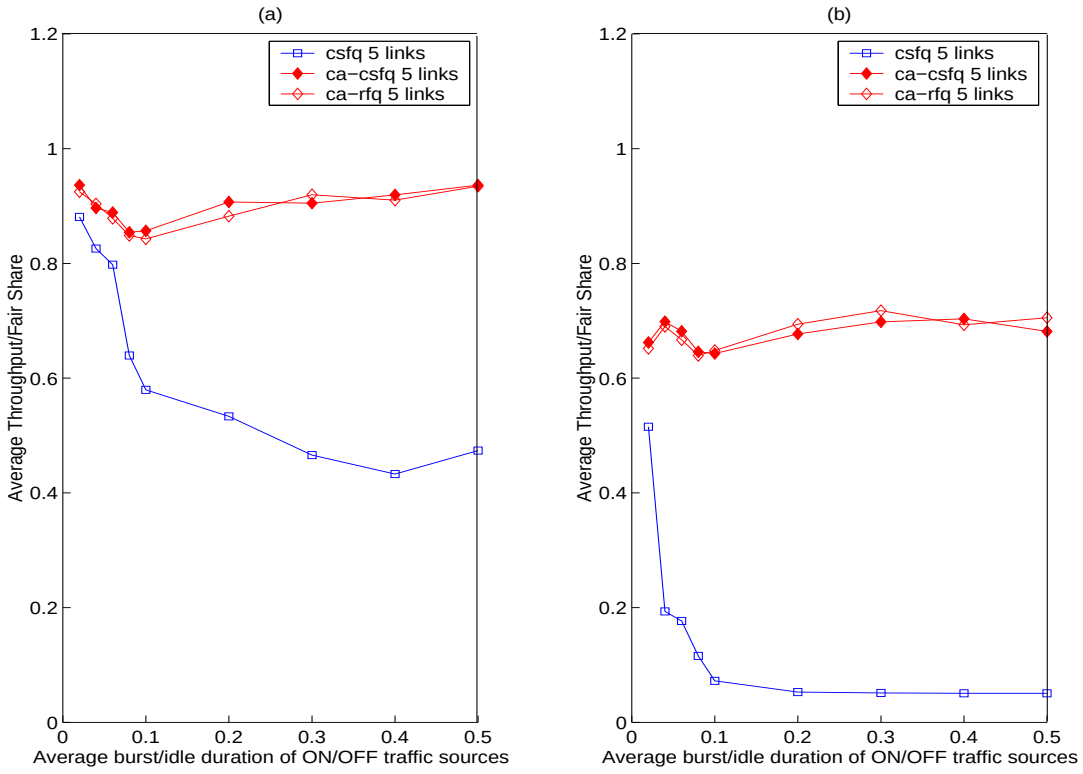

Figure 5.6: (a) Normalized throughput of a UDP flow going through 5 congested links. Cross traffic are ON/OFF sources whose average rate is twice the fair share. The burst and idle times vary between 20 msec to 0.5 sec. (b) The same plot as (a) but with the UDP flow being replaced by a TCP flow.

scheme, consisting of sending feedback packets back to the traffic source for rate control based on past flow rate values or past buffer occupancy values, or a combination of the two sets of values. In the past few years, these systems have been studied extensively with a recent emphasis on the use of a control theoretical approach [79], [82], [83].

There are however, two key differences between this proposed system and that of an end-to-end rate-based flow control system, such as the ATM ABR congestion control systems mentioned above:

(1) Firstly, the proposed system involves only one incoming traffic source because the SCORE network architecture aggregates all traffic at the core routers. On the other hand, ABR congestion control system requires per-flow state management of multiple traffic sources.

(2) Secondly, the proposed control systems are localized at the core routers. Feedback values used are obtained from the periodic sampling of F and buffer occupancy, which eliminates uncertain feedback delay and results in a more robust system. On the other hand, the end-to-end flow control system of ABR traffic involves the uncertain feedback delay due to the round-trip time of the feedback resource management cells [81]. This delay is normally very significant and can be uncertain resulting in stability issues.

Presently, besides CSFQ and RFQ, TUF [39] is the only other well-known SCORE/DPS fair queueing algorithm. TUF follows the same implementation framework of RFQ but uses a different rate representation and packet dropping algorithm. For the packet dropping algorithm, TUF discards the packet with the highest tag when the buffer is full. However, the complexity of searching the packet with the highest tag increases

linearly with the queue size. Depending on the size of the buffer, this search process may be more computationally intensive when compared to CSFQ, RFQ and the proposed CA-CSFQ and CA-RFQ. Note that TUF's packet dropping algorithm can also be replaced with the proposed system. Another aspect of TUF is that it takes into account the congestion responsive nature of flows and adjusts loss rates such that the average rates are equal. Unfortunately, this requires fairly accurate mathematical models of the end-to-end congestion responsive nature of traffic flows, which may not be readily available.

Another algorithm that aims to achieve the same objective is CHOKe (CHOose and Keep or responsive flows, CHOose and Kill for unresponsive flows) [84]. CHOKe is essentially an extension of RED [76] and like RED, CHOKe maintains two threshold values, min_{th} (minimum threshold) and max_{th} (maximum threshold) for the computation of drop probability. When the average queue size is bigger than min_{th}, each arriving packet is compared with a randomly selected packet in the buffer. If they have the same flow identifier, then both packets are dropped. Otherwise, only the arriving packet is considered for dropping, with a probability that depends on the average queue size. This drop probability is computed exactly as in RED. However, a comparison of simulation results of CHOKe [84] with the other SCORE / DPS fair queueing algorithms like CSFQ [37], RFQ [38] and TUF [39] showed that the flow isolation properties of CHOKe is significantly worse. The main reason is that CHOKe was meant for the simpler DiffServ [10] architecture that do not require an addition DPS packet label. Therefore, CHOKe works at a much coarser granularity and is unable to emulate the

performance of SCORE / DPS algorithms.

Recently, Hollot et. al. [85] did a control-theoretical analysis of the RED system and demonstrated that RED becomes less stable as the number of sessions decreases and the average session round trip time increases. He subsequently proposed the use of a proportional-integral controller to rectify this stability issue [86].

Athuraliya et. al. [87] also proposed a new active queue management algorithm, called Random Exponential Marking (REM), which measures congestion by a price quantity instead of a performance measure such as loss or delay. The price updating function, which is similar to equation (5.12), is computed based on a weighted function of the mismatches between the actual backlog and the desired backlog, and between the incoming traffic rate and link service rate. This price is then used to determine a dropping or marking probability. REM was shown to achieve high link utilization with negligible delays and buffer overflow regardless of the number of flows. However, REM does not address isolation issues between TCP and UDP flows.

5.6 Conclusion

This chapter presents a formal method for designing the packet dropping component of SCORE/DPS fair queueing algorithms, such as CSFQ and RFQ. Using a control theoretical approach, a generic control system is developed for the SCORE/DPS fair queueing core router dynamics. In particular, it is demonstrated how an optimal control approach can be used to design a stable system that allows for arbitrary control of the core router's performance. Compared to the original CSFQ and RFQ algorithms, the

resulting controller is simpler because there is no need to apply additional logic for saturated traffic conditions. Simulation results have been presented to show that the resulting controller yields better results than the original CSFQ algorithm.

Chapter 6

Conclusion and Future Work

In this chapter, this thesis is concluded by summarizing the contributions made and proposing some directions for future work.

6.1 Contributions

The main objective of this thesis is to develop a service differentiation architecture for the Internet that can resolve the two key insufficiencies of the current best-effort service paradigm: (a) Inefficient network resource utilization due to the lack of service differentiation and (b) Congestion unresponsive flows squeezing out congestion responsive flows due to the lack of flow isolation.

To keep the proposed architecture scalable and simple to implement, deploy and manage, a best-effort enhancement approach was adopted over a resource reservation approach. Another motivation is to maintain the flat rate type of commercial agreement between network operators and subscribers, widely believed to be the key factor underlying the rapid deployment of the Internet over the last few years.

The proposed **Delay-Rate Differentiated Services (DRDS)** architecture is one

that meets all the above requirements. It is built upon the IETF recommended DiffServ architecture and consists of two portions.

The first portion of the DRDS focuses on providing delay based service differentiation for classes of traffic aggregates. A Delay-Rate Differentiation (DRD) model, which refines on the Proportional Delay Differentiation (PDD) model, proposed by Dovrolis under the Proportional Differentiated Services (PDS) framework [13], is proposed. The DRD model is a combination of the PDD model with another proposed model, called the Generalized Minimum Queueing Delay (GMQD) model. Several packet scheduling algorithms emulating GMQD [15], [16], [17], and DRD are proposed and analyzed. For GMQD, two Packetized GMQD (PGMQD) algorithms, called Queue Length based Packetized Generalized Minimum Queueing Delay (QL-PGMQD) and Queueing Delay based Packetized Generalized Minimum Queueing Delay (QD-PGMQD), are proposed. For DRD, two Packetized DRD (PDRD) algorithms, called Queue Length based Packetized Delay Rate Differentiation (QL-PDRD) and Queueing Delay based Packetized Delay Rate Differentiation (QD-PDRD), are proposed.

From the simulation results, it was demonstrated that the class of Queueing Delay based packet scheduling algorithms, WTP, QD-GMQD and QD-PDRD, has better performance compared to the class of Queue Length based packet scheduling algorithms, BPR, QL-GMQD and QL-PDRD, but comes at the expense of computational complexity. To resolve this complexity issue, a novel approximation technique is demonstrated through the proposal of Scaled-Time Priority (STP) [69], [70], [71], which is an efficient approximation to Waiting Time Priority (WTP) [43]. The same technique is subse-

quently used to simplify QD-PGMQD and QD-PDRD, resulting in Scaled QD-PGMQD (SQD-PGMQD) and Scaled QD-PDRD (SQD-PDRD).

The second portion of DRDS focuses on providing flow isolation within each class of traffic aggregates. This portion leverages upon the class of core-stateless fair queueing algorithms proposed under the Stateless Core (SCORE) framework. Under this framework, Stoica proposed the Core-Stateless Fair Queueing (CSFQ) [37]. Unfortunately, CSFQ is known to perform badly under bursty traffic conditions, which can be attributed to its ad-hoc fair share estimation method. In view of this, a control-theoretical approach for fast and robust computation of the fair share value is proposed. This approach is used to enhance CSFQ [37] and its variant, Rainbow Fair Queueing (RFQ) [38], resulting in two new and improved algorithms, called Control-theoretical Approach to CSFQ (CA-CSFQ) [50], [51] and Control-theoretical Approach to RFQ (CA-RFQ) [51].

6.2 Future Work

In this section, three possible extensions to this thesis are explored.

(1) Loss Differentiation. This thesis focused only on delay differentiation. A possible extension will be to enhanced the proposed DRD model with a loss differentiation component [48], [49].

(2) Coordinated end-to-end multi-hop scheduling. Recently, several researchers have proposed schedulers that are capable of doing coordinated multi-hop scheduling, whereby downstream nodes have an opportunity to make up for excessive latencies due to con-

gestion at upstream nodes and vice versa [88], [89]. These algorithms exploit inter-node coordination to improve the end-to-end performance of traffic flows, and consequently, improve the efficiency and utilization of the network at large. Therefore, another possible improvement will be to design coordinated PGMQD and PDRD algorithms.

(3) End-to-end Feedback. CSFQ, RFQ, TUF, CA-CSFQ and CA-RFQ provide approximate local max-min fairness, but do not provide global max-min fairness [77]. A possible improvement to the proposed CA-CSFQ and CA-RFQ algorithms is to incorporate end-to-end feedback mechanisms [90]. It has been shown that enhancing CSFQ with end-to-end feedback mechanisms can lead to better network resource utilization and achieve approximate global max-min fairness.

Bibliography

[1] J. Postel, Internet Protocol, *IETF RFC 791*, DARPA, Sep 1981.

[2] V. Jacobson, Congestion Avoidance and Control, *Proc. ACM SIGCOMM*, pp 314-329, Aug 1988.

[3] J. Postel, User Datagram Protocol, *IETF RFC 768*, Information Sciences Institute, Aug 1980.

[4] J. Nagle, On packet switches with infinite storage, *IEEE Trans. Communications*, Vol 35, No. 4, pp 435-438, Apr. 1987.

[5] A.K.J. Parekh, R. Gallager, A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case, *IEEE/ACM Trans. on Networking*, Vol 1, No. 3, pp 344-357, June 1993.

[6] S.J. Golestani, A Self-Clocking Fair Queuing Scheme for Broadband Applications, *Proc. IEEE INFOCOM*, 1994.

[7] J.C.R. Bennett and H. Zhang, Hierarchical Packet Fair Queueing Algorithms, *IEEE/ACM Trans. on Networking*, Vol. 5, No. 5, Oct 1997.

[8] V. Firoiu, J-Y.L. Boudec, D. Towsley and Z.L. Zhang, Theories and Models for Internet Quality of Service, *Proc. IEEE*, Aug 2002.

[9] R. Braden, D. Clark and S. Shenker, Integrated Services in the Internet Architecture: An Overview, *IETF RFC 1633*, Jun 1994.

[10] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, An Architecture for Differentiated Services, *IETF RFC 2475*, Dec 1998.

[11] A.M. Odlyzko, Internet pricing and the history of communications, *Computer Networks, Elsevier Science*, Vol. 36, pp 493-517, 2001.

[12] I. Stoica, Stateless Core: A Scalable Approach for Quality of Service in the Internet, *PhD. Dissertion, CMU-CS-00-176*, 2000.

[13] C. Dovrolis, Proportional Differentiated Services for the Internet, *PhD. Dissertion*, 2000.

[14] L. Massoulie, J. Roberts, Bandwidth sharing: objectives and algorithms, *Proc. IEEE INFOCOM*, 1999.

[15] H. T. Ngin, C. K. Tham and W. S. Soh, A Multirate Scheduling Algorithm using Dynamic Programming for ATM Networks, *Proc. IEEE APCC/ICCS'98*, vol. 1, pp. 97-102, Nov 1998.

[16] H. T. Ngin, C. K. Tham and W. S. Soh, Generalised Minimum Queuing Delay: An Adaptive Multi-rate Service Discipline for ATM Networks, *Proc. IEEE INFOCOM'99*, vol. 1, pp. 398-404, Mar 1999.

[17] C. K. Tham, R. A. Lou and H. T. Ngin, Minimum Queuing Delay: An Adaptive Algorithm for Scheduling in a Dynamic Job Shop, *Proc. IEEE IECON'99*, Nov 1999.

[18] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, Resource ReSerVation Protocol (RSVP)–Version 1 Functional Specification, *IETF RFC 2205*, Sep 1997.

[19] S. Shenker, C. Partridge and R. Guerin, Specification of Guaranteed Quality of Service, *IETF RFC 2212*, Sep 1997.

[20] J. Wroclawski, Specification of the Controlled-Load Network Element Service, *IETF RFC 2211*, Sep 1997.

[21] P. Almquist, Type of service in the internet protocol suite, *IETF RFC 1349*, Jul 1992.

[22] S. Deering and R. Hinden, Internet Protocol, Version 6 (IPv6) Specification, *IETF RFC 2460*, Dec 1998.

[23] K. Nicholas, S. Blake, F. Baker and D. Black, Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers, *IETF RFC 2474*, Dec 1998.

[24] D. Grossman, New Terminology and Clarifications for Diffserv, *IETF RFC 3260*, Apr 2002.

[25] K. K. Ramakrishnan, Sally Floyd, and D. Black, The addition of explicit congestion notification (ECN) to IP, *IETF RFC 3168*, Sep 2001.

[26] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S.Davari, V.Firoiu and D. Stiliadis, An Expedited Forwarding PHB (Per-Hop Behavior), *IETF RFC 3246*, Mar 2002.

[27] J. Heinanen, F.Baker, W.Weiss and J. Wroclawski, Assured Forwarding PHB group, *IETF RFC 2597*, Jun 1999.

[28] V. Jacobson, K. Nichols and K. Poduri, An Expedited Forwarding PHB, *IETF RFC 2598*, Jun 1999.

[29] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski and E. Felstaine, A Framework for Integrated Services Operation over DiffServ Networks, *IETF RFC 2998*, Nov 2000.

[30] F. Baker, C. Iturralde, F. Le Faucheur and B. Davie, Aggregation of RSVP for IPv4 and IPv6 Reservations, *IETF RFC 3175*, Sep 2001.

[31] I. Stoica and H. Zhang, Providing Guaranteed Services Without Per Flow Management, *Proc. ACM SIGCOMM*, Aug 1999.

[32] Z.L. Zhang, Z. Duan and Y.T. Hou, Virtual Time Reference System: A Unifying Scheduling Framework for Scalable Support of Guaranteed Services, *IEEE J. Select. Area Commun*, Special Issue on Internet QoS, Vol. 18, No. 12 pp 2684-2695, Dec 2000.

[33] Z.L. Zhang, Z. Duan and Y.T. Hou, Fundamental Trade-offs in Aggregate Packet Scheduling, *Proc. IEEE ICNP*, Nov 2001.

[34] Z.L. Zhang, Z. Duan, L. Gao and Y.T. Hou, Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Service, *Proc. ACM SIGCOMM*, Aug 2000.

[35] Z.L. Zhang, Z. Duan and Y.T. Hou, On Scalable Design of Bandwidth Brokers, *IEICE Trans. Commun*, Vol. E84-B, No. 8, Aug 2001.

[36] I. Stoica and H. Zhang, LIRA: An Approach for Service Differentiation in the Internet, *Proc. NOSSDAV*, Jun 1998.

[37] I. Stoica, S. Shenker and H. Zhang, Core-Stateless Fair Queueing: Achieving Approximate Fair Bandwidth Allocations in High Speed Networks, *Proc. ACM SIGCOMM*, Sep 1998.

[38] Z.R. Cao, Z. Wang and E. Zegura, Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State, *Proc. IEEE INFOCOM*, 2000.

[39] A. Clerget and W. Dabbous, *T*UF: Tag-based Unified Fairness, *Proc. IEEE INFOCOM*, 2001.

[40] P. Hurley and J.-Y. L. Boudec, P. Thiran, and M. Kara, ABE: Providing a Low-Delay Service within Best Effort, *IEEE Network Magazine*, Vol. 15, No. 3, May 2001.

[41] V. Firoiu, X. Zhang and Y. Guo, Best Effort Differentiated Services: Trade-off Service Differentiation for Elastic Applications, *Proc. IEEE ICT*, Jun 2001.

[42] C. Dovrolis and P. Ramanathan, A Case for Relative Differentiated Services and the Proportional Differentiation Model, *IEEE Network*, Oct 1999.

[43] C. Dovrolis, D. Stiliadis and P. Ramanathan, Proportional Differentiated Services: Delay Differentiation and Packet Scheduling, *Proc. ACM SIGCOMM*, Sep 1999.

[44] Y. Moret and S. Fdida, A proportional queue control mechanism to provide differentiated services, *Proc. ISCIS*, Oct 1998.

[45] M. Tufail, G. Jennes and G. Leduc, A scheduler for delay-based service differentiation among AF classes, *Proc. IFIP TC6 ICBC*, pp. 93-102, Nov 1999.

[46] C.C. Li, S.L. Tsao, M.C. Chen, Y. Sun and Y.M. Huang, Proportional Delay Differentiation Service Based on Weighted Fair Queueing, *Proc. IEEE ICCCN*, Oct 2000.

[47] T. Nandagopal, N. Venkitaraman, R. Sivakumar and V. Bharghavan, Delay Differentiation and Adaptation in Core Stateless Networks, *Proc. IEEE INFOCOM*, Mar 2000.

[48] C. Dovrolis and P. Ramanathan, Proportional Differentiated Services, Part II: Loss Rate Differentiation and Packet Dropping, *Proc. IWQoS*, Jun 2000.

[49] U. Bodin, A. Jonsson and O. Scheln, On Creating Proportional Loss-Rate Differentiation: Predictability and Performance, *Proc. IWQoS*, pp. 372-386, Jun 2001.

[50] H. T. Ngin and C. K. Tham, A Control-Theoretical Approach for Fair Share Computation in Core-Stateless Networks, *Proc. IEEE QoFIS/ICQT 2002, Springer-Verlag LNCS, Berlin Heidelberg New York*, Oct 2002.

[51] H. T. Ngin and C. K. Tham, A Control-Theoretical Approach for Achieving Fair Bandwidth Allocations in Core-Stateless Networks, *Computer Networks, Elsevier Science*, Vol. 40, No. 6, pp. 727-741, Dec 2002.

[52] L. Kleinrock, Queueing Systems Volume II: Computer Applications, *John Wiley and Sons*, 1976.

[53] A. Odlyzko, Paris Metro Pricing: The Minimalist Differentiated Services Solution, *Proc. IWQoS*, Jun 1999.

[54] W.A. Gale and R. Koenker, Pricing interactive computer services, *Computer Journal*, Vol. 27, No. 1, pp 8-17, Oct 1984.

[55] R. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[56] D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Method*. Academic Press, NY, 1982.

[57] A. Demers, S. Keshav and S. Shenker, Analysis and simulation of a fair queueing algorithm, *Proc. ACM SIGCOMM*, Sep. 1989, pp.1-12.

[58] J.C.R. Bennett and H. Zhang, WF2Q: Worst-Case Fair Weighted Fair Queuing, *Proc. IEEE INFOCOM*, April 1996.

[59] L. Kleinrock, A Delay Dependent Queue Discipline, *Nav. Res. Log. Quart. 11*, pp. 329-341, 1964.

[60] A.K.J. Parekh, R. Gallager, A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case, *IEEE/ACM Trans. on Networking*, Vol 2, No. 2, pp 137-150, April 1994.

[61] J. Liebeherr, D. Wrege and D. Ferrari, Exact Admission Control for Networks with a Bounded Delay Service, *IEEE/ACM Trans. Networking*, Vol. 4, No. 6, Dec 1996.

[62] D. Wrege, J. Liebeherr, A Near-Optimal Packet Scheduler for QoS Networks, *Proc. IEEE INFOCOM*, 1997.

[63] P. Goyal, H. M. Vin, H.C. Cheng, Start-Time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks, *IEEE/ACM Trans. on Networking*, Vol. 5, No. 5, pp 690-703, October 1997.

[64] P. Goyal, Packet Scheduling Algorithms for Integrated Services Network, *Ph.D Dissertation*, Univ. of Texasat Austin, Dep. Computer Sci., Aug 1997.

[65] S. McCreary and K.C. Claffy, Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange, *http://www.caida.org/outreach/papers/*, 27 Sep 2000.

[66] S. Bodamer, A Scheduling Algorithm for Relative Delay Differentiation, *Proc. IEEE HPSR*, Jun 2000.

[67] J. Liebeherr and N. Christin, Rate Allocation and Buffer Management for Differentiated Services, *Computer Networks, Elsevier Science* Vol. 40(1), Special Issue on the New Internet Architecture, pp. 89-110, Sep 2002.

[68] N. Christin, J. Liebeherr, and T. Abdelzaher, A quantitative assured forwarding service, *Proc. IEEE INFOCOM*, Jun 2002.

[69] H. T. Ngin and C. K. Tham, Achieving Proportional Delay Differentiation Efficiently, *Proc. IEEE ICON 2002*, pp. 27-30 Aug 2002.

[70] H. T. Ngin and C. K. Tham, Achieving Proportional Delay Differentiation Efficiently, *Computer Communications, Elsevier Science*, Vol. 27, No. 2, pp. 153-161, Feb 2004.

[71] H. T. Ngin and C. K. Tham, Scaled Time Priority: An Efficient Approximation to Waiting Time Priority, to appear in *Computer Networks, Elsevier Science.*

[72] D. Ferrari and D. Verma, A scheme for real-time channel establishment in wide-area networks, *IEEE J. Select. Area Commun*, Vol. 8, No. 3, pp 368-379, April 1990.

[73] T. Stoica, H. Zhang, T.S. Eugene Ng, A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Services, *Proc. ACM SIGCOMM*, 1997

[74] E. Knightly and H. Zhang, Traffic Characterization and Switch Utilization Using Deterministic Bounding Interval Dependent Traffic Models, *Proc. IEEE INFOCOM*, pp. 1137-1145, Apr 1995.

[75] M. Shreedhar and G. Varghese, Efficient Fair Queueing Using Deficit Round Robin, *Proc. ACM SIGCOMM*, Sep. 1995.

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

[77] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1987.

[78] L. Benmohamed and S.M. Meerkov, Feedback control of congestion in packet switching networks: The case of a single congested node, *Control Group Rep. CGR-93-15*, Univeristy of Michigan, Nov 1993.

[79] L. Benmohamed and Y.T. Wang, A Control-Theoretical ABR Explicit Rate Algorithm for ATM Switches with Per-VC Queueing, *Proc. IEEE INFOCOM*, 1998.

[80] B.D.O. Anderson and J.B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice-Hall, N.J., 1990.

[81] The ATM Forum Traffic Management Working Group, Traffic Management Specification, Version 4.0, *The ATM Forum*, Apr 1996.

[82] E. Altman, T. Basar and R. Srikant, Robust Rate Control for ABR Sources, *Proc. IEEE INFOCOM*, Mar 1998.

[83] A. Kolarov and G. Ramamurthy, A Control-Theoretical Approach to the Design of an Explicit Rate Controller for ABR Service, *IEEE/ACM Trans. Networking*, Vol. 7, No. 5, pp. 741-753, Oct 1999.

[84] R Pan, B. Prabhakar and K. Psounis, CHOKe A stateless active queue management scheme for approximating fair bandwidth allocation, *Proc. IEEE INFOCOM*, Mar 2000.

[85] C. Hollot, V. Misra, D. Towsley and W. Gong, A Control Theoretic Analysis of RED, *Proc. IEEE INFOCOM*, pp. 1510-1519, Apr 2001.

[86] C. Hollot, V. Misra, D. Towsley and W. Gong, On Designing Improved Controllers for AQM Routers Supporting TCP Flows, *Proc. IEEE INFOCOM*, Apr 2001.

[87] S. Athuraliya, V.H. Li, S.H. Low and Q.H. Yin, REM: Active Queue Management, *IEEE Network*, pp 48-53, May/Jun 2001.

[88] M. Andrews and L. Zhang, Minimizing end-to-end delay in high-speed networks with a simple coordinate schedule, *Proc. IEEE INFOCOM*, Mar 1999.

[89] C. Li and E. W. Knightly, Coordinated Multihop Scheduling: A Framework for End-to-End Services, *IEEE/ACM Trans. Networking*, Vol. 10, No. 6, pp 776-789, Dec 2002.

[90] C. Albuquerque, B. J. Vickers and T. Suda, Network Border Patrol, *Proc. IEEE INFOCOM*, Mar 2000.

Appendix A

Proof of Proposition 3.1

Proposition 3.1. Given $\delta_{k-1}/\delta_k \geq 2$ for $1 < k \leq N$, $\delta_1 = +\infty$ and $r_1 > r_2 > \ldots > r_N$, then

$$\frac{\delta_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l} < \frac{\delta_k r_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l^2} \qquad \text{for} \quad k = 1 \tag{A.1}$$

and

$$\frac{\delta_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l} > \frac{\delta_k r_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l^2} \qquad \text{for} \quad k = 2, \ldots, N \tag{A.2}$$

Proof: For the proof of equation (A.1), the following condition is required

$$\frac{\delta_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l} < \frac{\delta_k r_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l^2} \qquad \text{for} \quad k = 1 \tag{A.3}$$

This is equivalent to showing:

$$\sum_{l=1}^{N} \delta_l r_l^2 < r_1 \sum_{l=1}^{N} \delta_l r_l \tag{A.4}$$

As $r_1 > r_2 > \ldots > r_N$, therefore, the above condition is always true.

For the proof of equation (A.2), the following condition is required

$$\frac{\delta_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l} > \frac{\delta_k r_k \bar{Q}_{agg}}{\sum_{l=1}^{N} \delta_l r_l^2} \qquad \text{for} \quad k = 2, \ldots, N \tag{A.5}$$

This is equivalent to showing:

$$\sum_{l=1}^{N} \delta_l r_l^2 - r_i \sum_{l=1}^{N} \delta_l r_l > 0 \qquad \text{for} \quad k = 2, \ldots, N \tag{A.6}$$

Consider the left-hand term, as $r_2 > r_3 > \ldots > r_N$,

$$\begin{aligned}
\sum_{l=1}^{N} \delta_l r_l^2 - r_k \sum_{l=1}^{N} \delta_l r_l &> \sum_{l=1}^{N} \delta_l r_l^2 - r_2 \sum_{l=1}^{N} \delta_l r_l \\
&> \delta_1 r_1 (r_1 - r_2) + \\
&\qquad \delta_3 r_3 (r_3 - r_2) + \ldots + \\
&\qquad \delta_N r_N (r_N - r_2) \\
&> (\delta_3 + \ldots + \delta_N) r_N (r_N - r_2) + \\
&\qquad \delta_1 r_1 (r_1 - r_2) \\
&> \delta_2 r_N (r_N - r_2) + \\
&\qquad \delta_1 r_1 (r_1 - r_2) \\
&> 0
\end{aligned}$$

The following points must be noted in the above derivation:

(1) The combined value of the negative terms $\delta_3 r_3 (r_3 - r_2) + \ldots + \delta_N r_N (r_N - r_2)$ is greater than $(\delta_3 + \ldots + \delta_N) r_N (r_N - r_2)$ because $r_2 > r_3 > \ldots > r_N$.

(2) The summation of the geometric progression

$$\begin{aligned}
\delta_3 + \ldots + \delta_N &= \delta_N \left(\frac{\frac{\delta_2}{\delta_N} - 1}{\frac{\delta_{N-1}}{\delta_N} - 1} \right) \\
&< \delta_N \left(\frac{\delta_2}{\delta_N} - 1 \right) \\
&< \delta_2 \\
&< \delta_1
\end{aligned}$$

because $\delta_{k-1}/\delta_k \geq 2$.

(3) $r_N(r_N - r_{N-2})$ is a negative term and $r_1(r_1 - r_2)$ is a positive term. As δ_1 is a positive infinite value, $\delta_2 r_N(r_N - r_2) + \delta_1 r_1(r_1 - r_2)$ will always be greater than zero. Hence, the proof completes.

□

Appendix B

Proof of Theorem 3.1

Proof: Before the start of the proof, a few notations are defined first.

λ_k: available bandwidth at the k^{th} stage.

g_k: return function for session k.

g_k^*: optimal return at the k^{th} stage.

where $k = 1, 2, \ldots, N$.

The theorem is proved using Dynamic Programming (DP), which is an inductive approach. A brief overview of how DP is done is described here for the reader's convenience. Referring to Figure B.1, the proof using DP is usually done in stages. A N dimensional problem in this case is broken up into N parts. It starts by finding the optimal return at the first stage, which is actually the optimal return for a single session. This solution is usually trivial.

Moving on to the second stage, the problem faced becomes two dimensional because it now consists of two sessions. At this stage, DP makes use of a transition function and

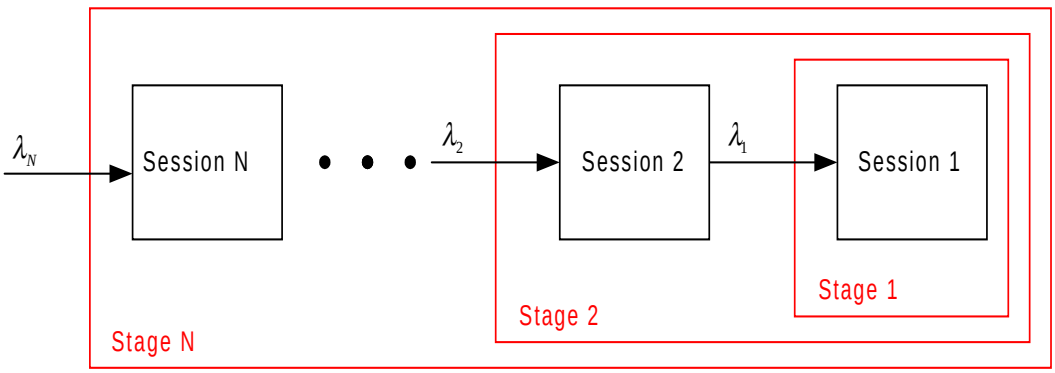

Figure B.1: Overview of the DP's approach to optimization.

a recursive formula to convert this two dimensional problem into a single dimensional problem. This simplifies the problem and makes it easier to obtain the optimal return at the second stage. Note that the optimal return at the second stage is the optimal return over two sessions, and not the optimal return for the second session only. In addition, the available bandwidth at the second stage is the total amount of bandwidth available to both the first and second sessions.

Similarly, at the third and later stages, the same procedure described earlier can be used recursively to reduce the dimensionality of the problem into a more manageable one dimensional problem. In other words, DP is a problem solving approach that recursively breaks a single N dimensional problem into N single dimensional problems. These N single dimensional problems are usually easier to solve.

The objective in $GMQD$ is to minimize the total weighted queueing delay for all sessions in one server. Hence, the objective function is

$$min \left[\frac{Q_1(t)w_1}{\phi_1(t)} + \frac{Q_2(t)w_2}{\phi_2(t)} + \cdots + \frac{Q_N(t)w_N}{\phi_N(t)} \right] \tag{B.1}$$

subject to the constraints that

(1) all service rates must be greater than or equal to zero and

(2) the sum of service rates is equal to the bandwidth of the server, $C(t)$.

However, as $GMQD$ is a fluid flow model, it is sufficient to consider the system at a single point in time. This formally reduces the objective function to

$$min \left[\frac{Q_1 w_1}{\phi_1} + \frac{Q_2 w_2}{\phi_2} + \cdots + \frac{Q_N w_N}{\phi_N} \right] \tag{B.2}$$

subject to the constraints

$$\phi_1, \phi_2, \ldots, \phi_N \geq 0 \tag{B.3}$$

and

$$\sum_{l=1}^{N} \phi_l = C, \ l = 1, 2, \ldots, N. \tag{B.4}$$

The first constraint, equation (B.3) is due to the fact that it is impossible to allocate negative service rates to any session. Similarly, the second constraint, equation (B.4) is due to the fact that the total service rates cannot exceed the bandwidth of the output link.

The transition function of this DP solution is an equation that relates the amount of resources (bandwidth) available at a particular stage to the amount of resources available and used up at the previous stage. In this case, it is

$$\lambda_{l-1} = \lambda_l - \phi_l, \ l = 1, 2, \ldots, N. \tag{B.5}$$

The recursive formula of this DP solution is an equation that relates the optimal return at a particular stage to the optimal return at the previous stage. In this case, it

is

$$g_k^*(\lambda_k) = \min_{0 \le \phi_k \le \lambda_k} [g_k(\phi_k) + g_{k-1}^*(\lambda_k - \phi_k)] \tag{B.6}$$

Both the transition function and the recursive formula are used to simplify expressions obtained in the subsequent derivations during the transition from one stage to another.

B.1 First Stage

First, consider the case where there is only one connected session to the output link. The optimal return function for the single session scenario is given as

$$g_1^*(\lambda_1) = \min_{0 \le \phi_1 \le \lambda_1} (\frac{Q_1 w_1}{\phi_1}) \tag{B.7}$$

Since the objective is to minimize the queueing delay of only one session, the obvious way is to allocate all the available bandwidth to that session That is $\phi_1 = \lambda_1$.

B.2 Second Stage

The next scenario is to minimize the total weighted queueing delay for two sessions. According to Bellman's principle of optimality [55], an optimal policy is made up of optimal sub-policies. Therefore, the optimal total return at the second stage is obtained by minimizing the sum of the return for the second session and the optimal return at the first stage. Hence,

$$g_2^*(\lambda_2) = \min_{0 \le \phi_2 \le \lambda_2} [g_2(\phi_2) + g_1^*(\lambda_1)] \tag{B.8}$$

There are three things to note here:

(1) $g_2(\phi_2)$ is the return function for the second session, given by $\frac{Q_2 w_2}{\phi_2}$. It is not the return function at the second stage, which is the total return for both the first and the second session.

(2) $g_1^*(\phi_1)$ is the optimal return function at the first stage, obtained when solving equation (B.7).

(3) Finally, the first session's service rate, ϕ_1 is the remainder of whatever bandwidth not taken up by the second session, i.e. $\phi_1 = \lambda_1 = \lambda_2 - \phi_2$. This is the transition function shown in equation (B.5). With this transition function, the two dimensional problem can be reduced to a single dimensional problem, which is easier to solve.

$$\begin{aligned} g_2^*(\lambda_2) &= \min_{0 \le \phi_2 \le \lambda_2} [\frac{Q_2 w_2}{\phi_2} + g_1^*(\lambda_2 - \phi_2)] \\ &= \min_{0 \le \phi_2 \le \lambda_2} [\frac{Q_2 w_2}{\phi_2} + \frac{Q_1 w_1}{(\lambda_2 - \phi_2)}] \end{aligned} \tag{B.9}$$

Let

$$G_2^*(\lambda_2) = \frac{Q_2 w_2}{\phi_2} + \frac{Q_1 w_1}{(\lambda_2 - \phi_2)}. \tag{B.10}$$

To minimize $G_2^*(\lambda_2)$ equate its first order differential to zero.

$$\begin{aligned} \frac{\partial G_2^*(\lambda_2)}{\partial \phi_2} &= \frac{Q_1 w_1}{(\lambda_2 - \phi_2)^2} - \frac{Q_2 w_2}{\phi_2^2} \\ &= 0 \end{aligned} \tag{B.11}$$

Solve for ϕ_2 and after simplification

$$\phi_2 = \frac{\sqrt{Q_2 w_2}}{\sqrt{Q_1 w_1} + \sqrt{Q_2 w_2}} \lambda_2 \tag{B.12}$$

Note that when solving the above quadratic equation, equation (A.11), the positive root for r_2 is taken.

To ensure that the solution found is a minimum, the second order differential is taken,

$$\begin{aligned}\frac{\partial^2 G_2^*(\lambda_2)}{\partial \phi_2^2} &= \frac{2Q_1 w_1}{(\lambda_2 - \phi_2)^3} + \frac{2Q_2 w_2}{\phi_2^3} \\ &> 0\end{aligned}$$

which is greater than 0 because $0 \leq \phi_2 \leq \lambda_2$. Hence, the solution for ϕ_2 is a minimum.

Making use of the transition function, $\phi_1 = \lambda_1 = \lambda_2 - \phi_2$, the value for ϕ_1 is obtained.

$$\phi_1 = \frac{\sqrt{Q_1 w_1}}{\sqrt{Q_1 w_1} + \sqrt{Q_2 w_2}} \lambda_2 \tag{B.13}$$

Dividing equation (B.12) by equation (B.13) and

$$\frac{\phi_2}{\phi_1} = \sqrt{\frac{Q_2 w_2}{Q_1 w_1}} \tag{B.14}$$

Substitute into $g_2^*(\lambda_2)$ at equation (B.9) and the optimal return function at the second stage becomes

$$g_2^*(\lambda_2) = \frac{(\sqrt{Q_1 w_1} + \sqrt{Q_2 w_2})^2}{\lambda_2} \tag{B.15}$$

B.3 Extending to Later Stages

Using the above procedure to recursively reduce the dimensions of the objective function, the optimal service rates and return functions can be obtained for the third, fourth and

later stages. Hence, by induction, at the N^{th} stage, the optimal service rate for session k and the optimal return function are given respectively by

$$\phi_k = \frac{\sqrt{Q_k w_k}}{\sum_{l=1}^{N} \sqrt{Q_l w_l}} \lambda_N \tag{B.16}$$

$$g_N^*(\lambda_N) = \frac{(\sum_{l=1}^{N} \sqrt{Q_l w_l})^2}{\lambda_N} \tag{B.17}$$

where $\lambda_N = C$. Include the time variable, t back into equations (B.16) and (B.17), and the proof is complete.

□

Appendix C

Proof of Theorem 3.2

Proof: From Theorem 3.1, the service rate of session k, $\phi_k(t)$ is given by

$$\phi_k(t) = \frac{C\sqrt{Q_k(t)w_k}}{\sum_{l=1}^{N}\sqrt{Q_l(t)w_l}},\ k = 1, 2, \ldots, N. \tag{C.1}$$

Given $\sum_{k=1}^{N} r_k = C$ and assuming the stable convergence of the adaptive GMQD system, then $\bar{\phi}_k \to \bar{r}_k$ at steady state. Therefore,

$$\bar{r}_k = \frac{C\sqrt{\bar{Q}_k w_k}}{\sum_{l=1}^{N}\sqrt{\bar{Q}_l w_l}},\ k = 1, 2, \ldots, N. \tag{C.2}$$

Dividing between two arbitrary sessions, k and l to obtain

$$\frac{\bar{r}_k}{\bar{r}_l} = \sqrt{\frac{\bar{Q}_k w_k}{\bar{Q}_l w_l}},\ l = 1, 2, \ldots, N.$$

Square both sides and rearrange the terms to obtain,

$$\bar{Q}_l = \left(\frac{w_k}{w_l}\right)\left(\frac{\bar{r}_l}{\bar{r}_k}\right)^2 \bar{Q}_k,\ l = 1, 2, \ldots, N.$$

Sum up Q_l for every session to obtain

$$\sum_{l=1}^{N}\bar{Q}_l = \left[\left(\frac{w_k}{w_1}\right)\left(\frac{\bar{r}_1}{\bar{r}_k}\right)^2 + \left(\frac{w_k}{w_2}\right)\left(\frac{\bar{r}_2}{\bar{r}_k}\right)^2 + \cdots + \left(\frac{w_k}{w_N}\right)\left(\frac{\bar{r}_N}{\bar{r}_k}\right)^2\right]\bar{Q}_k \tag{C.3}$$

Rearranging the terms to give

$$\bar{Q}_k = \frac{\bar{Q}_{agg}}{\sum_{l=1}^{N} \left(\frac{w_k}{w_l}\right) \left(\frac{\bar{r}_l}{\bar{r}_k}\right)^2} \tag{C.4}$$

where

$$\bar{Q}_{agg} = \sum_{l=1}^{N} \bar{Q}_l$$

Since $\bar{\phi}_k \rightarrow \bar{r}_k$ at steady state, D_k^{SS} is therefore, given by

$$\begin{aligned} \bar{D}_k &= \frac{\bar{Q}_k}{\bar{r}_k} \\ &= \frac{\bar{Q}_{agg}}{\bar{r}_k \sum_{l=1}^{N} \left(\frac{w_k}{w_l}\right) \left(\frac{\bar{r}_l}{\bar{r}_k}\right)^2} \end{aligned} \tag{C.5}$$

This completes the proof.

□