12-13-2002

Scheduling for Proportional Differentiated Services on the Internet

Manimaran Selvaraj

SCHEDULING FOR PROPORTIONAL DIFFERENTIATED SERVICES ON THE INTERNET

By

Manimaran Selvaraj

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2002

SCHEDULING FOR PROPORTIONAL DIFFERENTIATED SERVICES ON THE INTERNET

By

Manimaran Selvaraj

Approved:

Georgios Y. Lazarou Assistant Professor of Electrical and Computer Engineering (Major Professor)	Nicholas Younan Professor of Electrical and Computer Engineering Graduate Coordinator (Committee Member)
Robert J. Moorhead Professor of Electrical and Computer Engineering (Committee Member)	A. Wayne Bennett Dean of the College of Engineering

Name: Manimaran Selvaraj

Date of Degree: December 13, 2002

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: Dr. Georgios Y. Lazarou

Title of Study: SCHEDULING FOR PROPORTIONAL DIFFERENTIATED SERVICES ON THE INTERNET

Pages in Study: 59

Candidate for Degree of Master of Science

Proportional Differentiated Services can be provisioned in terms of bandwidth, delay, or packet loss. Several studies contributed schedulers and packet droppers that achieved proportional bandwidth, delay, or loss differentiation. However, all these schemes differentiated in terms of only one of the three metrics. A simple, unified, scalable, and robust scheme to simultaneously control all three metrics was felt important.

By controlling just delay and packet loss, proportional differentiation can be achieved in terms of all three metrics. A robust adaptive scheduler for proportional delay differentiation services is presented. Proportional services are further policed by a class based packet dropper. The combination of the adaptive scheduler and the packet dropper treats different traffic classes proportionally in terms of all three metrics. Simulation experiments show that regardless of the network traffic characteristics, our scheme can effectively differentiate services in terms of bandwidth, delay, and loss simultaneously.

DEDICATION

This work is dedicated to the Telecommunication and Information Technology Laboratory (TITL).

ACKNOWLEDGMENTS

This work would not have been possible without the relentless guidance and support of my major advisor Dr. Georgios Lazarou. His belief in my capacity really helped me complete this work. It was a rewarding experience to work under his guidance for more than an year. My sincere thanks are due to him.

I thank Dr. Robert Moorhead, my committee member, for his patient and timely guidance on my career path. I also thank him for really understanding my career motives and directing me to Dr. Lazarou for this work.

I thank Dr. Nicholas Younan, Graduate Coordinator and my committee member, and the Department of Electrical and Computer Engineering for all the financial support, office space, computer, and all the other help.

I also thank my friends and members of TITL (Gomathi Anandan and Dong Zheng in particular) who helped me at various stages of this work.

Finally, I would like to say that the time spent in Simrall Hall will be the most memorable of my student life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

AF	Assured Forwarded PHB
$DiffServ$	Differentiated Services
EF	Epedited Forwarded PHB
HPD	Hybrid Proportional Delay
$IETF$	Internet Engineering Task Force.
$IntServ$	Integrated Services
IP	Internet Protocol
ISP	Internet Service Provider
PAD	Proportional Average Delay
PDD	Proportional Delay Differentiation
PHB	Per-hop Behavior
QoS	Quality of Service
RED	Random Early Detection
RIO	RED with In/Out packets
$RSVP$	Resource Reservation Protocol
SLA	Service Level Agreement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WFQ	Weighted Fair Queing
WTP	Waiting Time Priority

CHAPTER I

INTRODUCTION

The Internet which originally started as a means of moving files from one computer to another has now evolved into a global communications network. The number and variety of applications on the Internet has grown in leaps and bounds. The IP[1] based Internet is now serving users ranging from ordinary home users to huge corporations. In spite of this tremendous growth in usage, the Internet architecture has under gone very little change. The Internet is still based on the best-effort service model. In this model, all data packets are treated equally and the network tries its best to ensure reliable delivery. This design is very simple and easily scalable, but it offers no guaranteed delivery of packets.

The huge success of the best-effort service model can be attributed to the underlying transmission mechanism - Transmission Control Protocol (TCP)[27]. During the early stages of the Internet's development, most of the applications used TCP as their transmission mechanism for reliable delivery. Only a few applications were using unreliable transmission layer protocols such as User Datagram Protocol (UDP) . Traffic sources that use TCP to transmit their data regulate their sending rate according to the perceived level

[1]IP Internet Protocol is the main data transfer protocol based on which the Internet is built

of congestion within the network [26]. Thus, these self-regulating TCP sources appear to share the network among themselves.

With the emergence of new applications like multimedia, voice, video, and fax over the Internet, this sharing of the network among the users is becoming increasingly unrealistic. For example, during congestion, a greedy UDP traffic source may continue to send data at the same rate, while the TCP sources would reduce their sending rate. Thus, the non-responsive UDP source gets a higher share of the network resources than the congestion responsive TCP sources. This indicates that the best effort model can not distribute resources in a fair manner in the near future. Thus, all the new applications, which are revolutionizing the communication industry, demand a better than best effort service. Moreover, in the future, businesses will place their mission-critical data, voice, and multimedia applications on the public Internet only if they are guaranteed a secure, predictable, and measurable service. Such demands can not be met by the best effort service model.

Today's Internet supports a much bigger number of users than ten years back. Very soon this increase would result in the full utilization of the global telecommunication network's bandwidth (physical carrying capacity). Therefore, the Internet must evolve into a new form using a new architecture to support the increased number of users and traffic load. This new architecture must not only be scalable (ability to handle expansive network growth), but must also support the older and existing applications.

The Internet service providers (ISP) consider these demands as a source of revenue. By offering service guarantees, security, and predictability, the ISPs can attract business customers with mission-critical traffic. In order to do this, ISPs have to classify and prioritize traffic according to the application's specific needs. In other words, the Internet must be able to discriminate users through the offered service quality. For example, the quality discrimination must be in such a way that users willing to pay more should receive a better service.

The need for such quality assurance, service guarantee, and reliability drove the Internet Engineering Task Force (IETF) to define two Quality of Service (QoS) solutions - Differentiated Services (DiffServ) and Integrated Services (IntServ). Of these two methods, the DiffServ gained popularity due to its inherent scalability, ease of implementation, and reduced operation complexity at the core of the Internet. These two solutions are described in detail in the next chapter.

In this thesis, a robust adaptive scheduler[2] for proportional delay differentiation services is presented. Proportional services are further policed by a class based packet dropper[3] which works along with the adaptive scheduler. The combination of the adaptive scheduler and the packet dropper treats different traffic classes proportionally in terms of all three QoS metrics: bandwidth, delay, and packet loss. We show through simulation experiments that regardless of the network traffic characteristics, the adaptive scheduler combined with the packet dropper can effectively differentiate services in terms of delay, bandwidth and loss simultaneously.

[2]Scheduling is the process of deciding which packet to send first in a system of multiple queues.

[3]A packet dropper drops packets at routers when congestion occurs.

1.1 The Problem Statement

In a DiffServ network, the quality of service (QoS) is measured in terms of [8] [12]:

1. Bandwidth - the maximum data transfer rate possible between the source and the destination,
2. Delay or Latency - the time a packet takes to traverse from the source to the destination, and
3. Reliability or Packet loss - the average error rate of the transmission medium.

The QoS solution is largely aimed at controlling one or more of these metrics for a particular data communication session between a sender and a receiver. In a DiffServ network[4], these performance metrics are controlled by packet schedulers and droppers.

DiffServ can be provided in an absolute or relative manner. In the former approach, DiffServ strives to achieve IntServ like performance guarantees; whereas in the latter, a traffic class is treated *relative* to another traffic class. The relative service differentiation has several advantages over the absolute service differentiation. The absolute service differentiation makes use of several mechanisms including, admission control[5], bandwidth brokerage[6], and resource reservation. Hence like IntServ, the absolute service differentiation suffers from scalability issues. On the other hand, the relative differentiation is much

[4]A DiffServ network is one that employs the DiffServ solution to provide QoS

[5]Admission Control is the decision process of whether to accept a request for resources

[6]A bandwidth broker is an agent responsible for allocating preferred service to users as requested and for configuring the network routers to the correct forwarding behavior for the defined service.

simpler. It is achieved through the use of packet schedulers and packet accept/discard rules. In the relative differentiated approach, Internet traffic is grouped into N finite number of classes. *Class i* gets a better or at least no worse service than *class i-1*.

Three scheduling algorithms were proposed in [5][7] to achieve a proportional (relative) delay differentiation between different traffic classes. Several scheduling algorithms [9][18][19][20] were later proposed based on [5][7]. A proportional loss differentiation technique was proposed in [6]. Soetens *et. al.* [25] made use of a version of the widely famous Random Early Detection (RED) algorithm [14] to achieve a relative bandwidth differentiation between individual TCP flows.

However, all these algorithms provide a relative service differentiation in terms of only one of the three service metrics (delay, loss or bandwidth). Although the work in [5][6][7] presents schemes to achieve a proportional delay as well as loss differentiation, a study of co-existence of the loss and delay differentiation schemes was not performed. Many of these schemes were not robust to handle traffic under various network conditions. Moreover, complex off-line computations [9] or computationally demanding algorithms [18] were needed for some of the schemes. Also, experiments conducted by Chen *et. al.* [3] demonstrated that due to the burstiness of the web traffic, DiffServ networks based on the Weighted Fair Queuing (WFQ) packet scheduling scheme cannot achieve the desired QoS guarantees.

The Internet has rapidly evolved into a heterogenous network. Internet applications greatly differ in QoS requirements. The DiffServ technique reduces this heterogeneity to a small extent by combining applications with similar QoS requirements into traffic classes. Nevertheless, in a global network like the Internet, a certain class of traffic may request strict delay guarantees, while another class would not tolerate excessive packet loss rate or inadequate bandwidth. In such networks, a simultaneous control of all QoS performance metrics is very much essential.

Since the Internet is a global network, three different schemes to control each of the three performance metrics independently would increase the complexity at every router. Moreover, Internet traffic load fluctuates greatly. Any scheme that controls the performance metrics must be very robust and tolerant to changing network conditions. Hence, a unified, simple, and robust technique, that simultaneously controls all the above three QoS metrics is required for the Internet.

To summarize, we address the following issues in this thesis:

1. A need for the simultaneous control of all three (bandwidth, delay, and packet loss) QoS performance metrics.

2. Such a scheme must be very simple to implement.

3. Although simple in nature, the scheme must be robust to handle the highly varying Internet traffic loads.

4. For a network that is as wide as the Internet, the scheme must be highly scalable.

5. And finally, the scheme must work extremely well in a heterogeneous network like the Internet.

Using our proposed simple scheme, we show that a simultaneous control of all three performance metrics is possible. The simulation experiment results also show that our scheme is robust.

1.2 Summary of Main Contributions

In this thesis, we propose a novel technique to simultaneously achieve proportional bandwidth, delay, and loss differentiation. We achieve our goal by developing a robust adaptive delay scheduler and integrating it with a class based packet dropper. The main contributions of this work are the following:

1. We developed a novel scheduler to achieve proportional delay differentiation between traffic classes. The scheduler adapts to the delay difference between classes.

2. Unlike other techniques [9] [18], our scheduler does not involve any complex computations. Hence, it is very simple.

3. We propose the use of multi-class RED [14][24] to achieve a proportional bandwidth differentiation. The proposed dropping scheme works in conjunction with the scheduler. The selective dropping and delaying of packets is reflected in the throughput achieved by the different classes.

4. Our packet dropper drops packets belonging to different classes proportionately. This results in proportional loss differentiation. Thus, our scheme simultaneously attains proportional bandwidth, delay, and loss differentiation.

5. Based on its class, the packet receives a particular scheduling (queuing) treatment and drop precedence. Thus, the adaptive scheduler and the class based dropper work together as a *single unit* in controlling the packet delay and bandwidth.

6. Our scheme does not depend on any external parameters like network load. Hence, irrespective of the network characteristics, our scheme can result satisfactory QoS.

To summarize, using our scheme, we show that by just manipulating the packet delay and loss, all three QoS metrics can be controlled in a robust manner. Hence, a proportional bandwidth, delay, and loss differentiation can be achieved simultaneously. Moreover, our technique involves much less complexity than techniques that have similar aims.

The rest of this thesis is organized as follows. In chapter II, we present the fundamentals of the two QoS provisioning methods. We discuss in brief, the IntServ and DiffServ Internet architectures, followed by the latter's advantages over the former. We then present a short list of requirements that led to the development of proportional DiffServ. Finally, we list several related works on the proportional DiffServ model.

Chapter III describes our proportional DiffServ technique. In this chapter, we present the theory and mathematics of our adaptive proportional delay scheduler and the class based packet dropper. We describe in detail the design of the scheduler and packet dropper. We also describe the initialization, parameter settings, and working of the two schemes.

In chapter IV, we present the performance evaluation methodology. We define the simulation model used to test our proposed proportional DiffServ scheme. The simulation set up, traffic sources used, and measurements made are all described in this chapter. Finally, we present results from the many simulation experiments that were performed to validate our claim that all three performance metrics can be simultaneously controlled.

In the final chapter V, we present our conclusions and discussions. We also suggest several ways in which this work can be enhanced further in other directions.

CHAPTER II

BACKGROUND AND RELATED WORK

2.1 QoS Basics

QoS refers to a set of rules or techniques that help the network administrators use the available network resources optimally to manage the effects of congestion [1] and to treat the applications according to their needs. Thus, the applications receive network resources in a controlled manner. As mentioned earlier, QoS provisioning can be achieved by using either the IntServ or the DiffServ mechanism. Numerous research experiments were performed using these two techniques to provision QoS in a network. In the rest of this chapter, we explain the fundamental principles of these two QoS provisioning methods. We then present several reasons that led to the success of the DiffServ architecture. It is then followed by the requirements that gave birth to the proportional DiffServ model. Finally, we list several works that contributed to the proportional DiffServ model.

2.1.1 IntServ Basics

The IntServ framework provides end-to-end QoS for each and every packet flow[1]. The end-to-end QoS provisioning is realized by pre-determining traffic path and pre-allocating

[1]A flow is a single stream of packets with common source and destination addresses, and port numbers

network resources for each and every flow. The IntServ achieves this by the integrated operation of four components [29]: signaling scheme, admission control routine, packet classification[2], and packet scheduling.

The signaling scheme currently used is the Resource Reservation Protocol (RSVP) [29]. The applications that require a preferential treatment set up paths and reserve resources in advance using RSVP (before actually transmitting any data). The admission control routines present at the intermediate routers along the path determine as to whether a request for resource could be granted or denied. The classifier verifies each packet's header and then forwards the packet to the appropriate packet queue. The packet scheduler then decides or selects the packet to be serviced so that its QoS requirements are met.

The IntServ architecture provides only two new classes of service [22]. Apart from supporting the already existing best effort service, the IntServ offers a guaranteed-service class and a controlled-load service class. In guaranteed-service class the service provider offers the user a strict service level bound. On the other hand, in the controlled-load service class, the ISP only promises the user a service quality level.

2.1.2 DiffServ Basics

DiffServ is based on the idea that all the traffic flows can finally be grouped into a finite number of traffic classes. Individual traffic flows with similar QoS requirements are com-

[2]Classification is the process of sorting packets based on the content of the packet headers.

bined together to form traffic aggregates (classes). The packet class is identified by a short label in the IP header called the DiffServ code point [3] (DSCP).

Whenever a host or edge router sends data into a DiffServ network, they first mark every packet with the appropriate DSCP value. The DSCP value is decided by a complex packet classification procedure. One or more entries of the packet (IP) header is used in this decision making process. In the DiffServ network, only the edge routers do this packet classification and thus the complexity is pushed to the network boundary.

The intermediate routers (routers internal to the DiffServ network) treat the received packets based on the short DSCP value alone[4]. In other words, the queuing behavior to be applied on the packet is selected based on the DSCP value. This queuing behavior is called the per-hop behavior[5] (PHB). The PHBs may differ at each and every hop (or router).

At present, four types of PHBs have been defined and have been standardized [2]. They are the default PHB, the class-selector PHB, the expedited forwarding (EF) PHB and the assured forwarding (AF) PHB. The default PHB provides the same kind of service as the existing best effort model. The Class-selector PHB offers seven different queuing behaviors, with increasing timely packet forwarding probability. The EF PHB offers premium end-to-end services consisting of low loss, low latency, low jitter, and assured bandwidth [15]. The AF PHB offers different levels of forwarding assurances for the received pack-

[3]The DSCP is a 6 bit field spanning the type of service (TOS) field and IP precedence field of the IP header [1].

[4]In the current Internet, the routers treat packets based on a full IP address search

[5]PHB defines how an individual router will treat a packet when sending it over the next hop through the network [2].

ets [13]. It delivers IP packets in four independently forwarded AF classes. Within each AF class, the IP packet can further be assigned to one of the three levels of drop probability or precedence.

Service level agreements (SLA) between the customer and service provider decides the type of service provided. The SLAs specify the available resources and the class of service offered by the service provider. The SLAs also restrict the amount of traffic allowed in each class. The SLAs are defined at the boundaries where the user submits traffic to the DiffServ network's ingress router. Any traffic that exceeds the SLA will be treated as a best-effort traffic. The SLAs also contain the rules that the ingress routers must follow while classifying, policing [6], and shaping [7] traffic. Whenever a packet leaves one DiffServ domain and enters another DiffServ domain, the DCSP values are remarked. The rule for this remarking procedure is derived from the SLA between the two DiffServ domains.

2.1.2.1 The Winning QoS Solution: DiffServ

A major difference between the two QoS provisioning architectures is that the DiffServ is a per-aggregate mechanism, while the IntServ is a per-flow mechanism. Since the number of service classes is very small in the DiffServ architecture, the amount of the state information held in the routers is small. Whereas, in the IntServ network, the routers have to store state information for every flow. Moreover, the DiffServ architecture deals with

[6]Policing is the process of handling out-of-profile traffic i.e., excess traffic [29].

[7]Traffic Shaping is the process of delaying packets within a traffic stream to cause it to conform to some defined traffic profile [29].

aggregates rather than individual flows. In order to serve these traffic aggregates, DiffServ does not need any signalling schemes like the way IntServ does. Thus, the less state information at the routers and the absence of a signalling scheme greatly increases the scalability of the DiffServ architecture.

The DiffServ architecture does not require the intermediate routers in the path to do complex packet classification (full IP header based) or reserve resources for each and every flow. Packet classification can be limited to the boundary routers present between the user and the service provider. The intermediate routers just have to do a simple packet forwarding based on the DSCP value. Hence, the core routers investigate the DSCP byte alone and forward the packets very quickly. This is not the case in the IntServ architecture where each and every router has to respond to resource reservation messages and also classify packets.

Another advantage of the DiffServ architecture is interoperability. DiffServ networks can be built on top of a set of independent DiffServ domains, each deploying an independent set of PHBs. Interoperability is achieved through specific functions at the boundaries between different domains like SLAs, PHB mapping, traffic shaping and policing.

Thus, for the above mentioned several reasons, the DiffServ architecture has emerged as the winning solution to provide QoS for the Internet. Many network equipment manufacturers are now actively looking into DiffServ as a viable solution for the future networks.

2.2 The Origin of Proportional Differentiation Mechanism

With the ever increasing number of Internet based applications, the need for service differentiation is being increasingly felt important. Many ISPs are now beginning to make use of techniques that differentiate one application from the other. This differentiation is necessary due to several issues such as QoS provisioning and pricing policies.

As mentioned earlier, the QoS offered to the applications can be measured in terms of one or a combination of the three performance metrics, namely, bandwidth, delay, and packet loss rate. On a monetary basis, a network user who pays more will expect a lower delay level and packet loss rate than a user who pays less. Likewise, from a performance point of view, a real time video transmission must experience lower delay levels than an ordinary email or file transfer. Thus, from an application point of view, traffic ought to be differentiated in terms of the delay and packet loss, since they differ in delay and loss requirements. This necessitates the need for a proportional delay and loss differentiation.

Large business traffic, IP based voice traffic, and video traffic are some of the huge opportunities identified by service providers in recent years. ISPs can attract these kind of customers by offering different levels of service. A higher priced service needs to have more bandwidth than a lower priced service. Hence, ISPs need to handle bandwidth more efficiently. In other words, bandwidth will also have to be *relatively* shared between applications. The service providers can also beat their competition by offering multiple classes of better than best effort services which differ in bandwidth. Thus, a bandwidth differentiation is needed in this case. Hence, the service offered must be differentiated in terms of all three (bandwidth, delay, and loss) quality metrics.

The Proportional DiffServ model was first proposed in [5][7][6]. Two independent versions of the model, namely, proportional delay differentiation and proportional loss differentiation were proposed. These models were the first in the proportional differentiation area and it served as a motivation for a number of works which proposed ways to proportionally differentiate service in terms of one of the three service metrics. Our work was also motivated by this trend setting proportional DiffServ model.

2.3 Related Research

The Proportional Delay Differentiation (PDD) model proposed in [5][7] paved the way to a whole new area of Internet research. In this model, the ratio of the overall long term average delay, $\overline{d}$, experienced by two different traffic classes *i* and *j* is equal to the ratio of their corresponding delay differentiation parameters (DDP) δ:

$$\frac{\overline{d}_i}{\overline{d}_j} = \frac{\delta_i}{\delta_j} \quad (i, j = 1...N) \tag{2.1}$$

The delay differentiation parameters $\{\delta_i\}$, are ordered as $\delta_1 > \delta_2 > ... > \delta_N > 0$, so that the higher classes experience less delay than the lower classes.

The work in [5] and [7] proposed to achieve proportional delay differentiation through the use of packet schedulers. Three schedulers were addressed and their performance was compared. The schedulers are the proportional average delay scheduler (PAD), the waiting time priority scheduler (WTP), and the hybrid proportional delay scheduler (HPD).

The PAD scheduler aims to equalize the normalized average delay among all classes. The normalized average delay is given by $\tilde{d}_i = \overline{d_i}/\delta_i$. The PAD scheduler selects the queue with the maximum normalized delay from a set of backlogged classes. The normalized average delay in this case is given as:

$$\tilde{d}_i(t) = \frac{1}{\delta_i}\frac{S_i}{P_i} \tag{2.2}$$

where S_i is the sum of the queuing delays of all the *class i* packets that have already been serviced at the current time *t*, and P_i is the corresponding count of *class i* packets that have been dequeued.

The WTP scheduler, which was originally studied by Kleinrock [17] under the name *Time dependent Priorities*, works to minimize the normalized head waiting times of different classes. The normalized head waiting time is given as:

$$\tilde{w_i}(t) = \frac{w_i(t)}{\delta_i} \tag{2.3}$$

where $w_i(t)$ is the waiting time of the head packet of a class or queue. The waiting time is measured as the difference between current system time and the time when a particular packet entered the queue. WTP provides a relative delay differentiation between successive packet departures and hence WTP works correctly even in short time scales. WTP's major drawback is that it fails to proportionally delay differentiate when the system utilization is low. Thus, at heavy loads WTP works perfectly well, but when load decreases, WTP fails to delay differentiate. In other words, WTP fails to delay differentiate bursty traffic where load fluctuates.

The higher system utilization requirement of WTP was overcome in two separate works - [9] and [18]. Both the approaches attempt to make WTP adaptive to varying loads. The adaptive approaches derive a feasible load distribution range for a given set of delay differentiation parameters (DDP) or class weight. Different approaches are proposed to calculate DDPs for any given load conditions. While the scheme in [18] made use of the Gauss-Seidel iterative method, [9] opted for an off-line DDP computation method followed by a table lookup procedure.

The HPD scheduler was an attempt to design a packet scheduler which had the best features of both PAD and WTP. The corresponding normalized average delay for HPD is given as:

$$\tilde{h}_i(t) = (g)\tilde{d}_i(t) + (1 - g)\tilde{w}_i(t) \tag{2.4}$$

where *'g'* is the HPD parameter, $\tilde{w}_i(t)$ is the normalized head waiting time as defined by (2.3), and $\tilde{d}_i(t)$ is the normalized average delay as defined by (2.2).

In general, the WTP and HPD schedulers perform better than the PAD. PAD is able to meet the PDD model only when the delay differentiation parameters are available; whereas, WTP works only under heavy loads. The selection of the HPD parameter *'g'* plays a very significant role in HPD's performance. Under heavy load conditions, the value of *'g'* does not affect the performance of HPD, since both PAD and WTP work well under heavy loads. But when the utilization decreases, the value of *'g'* must be close to 1 so that HPD works more like PAD [5].

Weighted fair queuing (WFQ) algorithm was used in [19] to achieve a proportional delay differentiation. In this scheme, an extended version of WFQ is used to achieve proportional delay differentiation. The scheme controls the delay difference between the classes by dynamically adjusting the weight of each class.

Dovrolis in [6] proposed a proportional loss differentiation mechanism. The technique was based on packet loss counters. So, the drop probability was decided based on the ratio of the loss count value of one class with respect to the other. [20] proposed yet another technique to achieve proportional loss differentiation between classes. In [20] a buffer management scheme was utilized to proportionally drop packets.

A relative bandwidth differentiation between TCP micro-flows was achieved in [25] by making use of the weighted version of RED[8], called WRED [24]. The authors made use of WRED to achieve a per-flow relative loss differentiation. They further proposed that a relative bandwidth differentiation could be achieved by a combination of a relative loss and a relative delay differentiation of the TCP micro-flows. By controlling packet loss rate and delay, a relative bandwidth differentiation was achieved between TCP micro-flows. The relative bandwidth differentiation between flows can be represented as:

$$\frac{BW_i}{BW_j} = \frac{\beta_i}{\beta_j} \tag{2.5}$$

[8]Random Early Detection [14]

where BW_i is the bandwidth or throughput achieved by flow i, and β_i is the bandwidth differentiation parameter of flow i.

To summarize, several algorithms were proposed to control the bandwidth, delay, or loss rate. But none of the technique simultaneously controlled more than one performance metric. Moreover, several schemes were also computationally demanding.

CHAPTER III

PROPORTIONAL DIFFERENTIATION MECHANISMS

As mentioned earlier, we can achieve simultaneous proportional bandwidth, delay, and loss differentiation by controlling the packet delay and loss factor alone. Our proposed adaptive scheduler provides a proportional delay differentiation, while the class based packet dropper proportionally loss differentiates. The scheduler and the packet dropper work in tandem to produce the desired bandwidth, delay, and loss differentiation. In this chapter, we present the design and working principles of these two schemes.

3.1 Proportional Delay Mechanism

We attain proportional delay differentiation between traffic classes by using an extended version of the HPD scheduler proposed in [5][7]. Motivated by the works in [9], [10], and [18] we made the HPD packet scheduler adaptive, so that the scheduler maintains the desired delay differentiation ratio under all network conditions. Numerous simulation experiments were performed with this new scheduler and the *adaptiveness* in the HPD scheduler helped to maintain the desired proportional delay differentiation. Reiterating our claims again, unlike other adaptive approaches [10][18], our adaptive approach proposed here is much simpler, robust, and it does not depend on the network load.

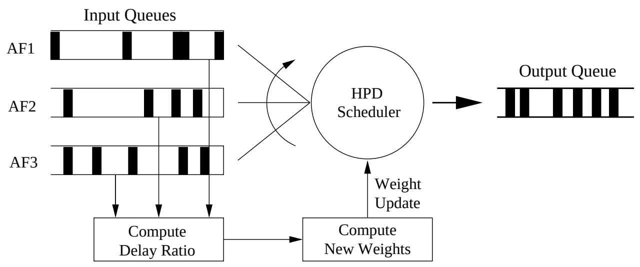

Figure 3.1 Adaptive HPD

3.1.1 Adaptive HPD

The work conducted by Dovrolis [7] was the first to define the proportional delay differentiation model and it gave examples of the environments in which the model worked. But, the schedulers mentioned in [7] do not achieve a proportional delay differentiation under low and medium system utilization. High system utilization is an important requirement of the model. Moreover, the schedulers PAD, WTP or HPD are not able to maintain a proportional delay differentiation strictly under various network conditions.

We propose a major addendum to the HPD scheduling scheme by adding a feedback component to the scheduler. We call this the adaptive HPD (AHPD) scheduler and is depicted in Fig. 3.1. As observed in the figure, in this proposed adaptive scheduler, the actual delay ratio between two traffic classes is periodically monitored and the delay ratio is computed. Using this calculation, the class weights are changed in such a way that the delay ratio is always maintained at the desired value.

We begin the design of our adaptive scheduler from the HPD scheduling scheme. Equation (2.4) which gives the normalized average delay for the HPD scheduler is repeated here for convenience. The HPD normalized average delay is given as:

$$\tilde{h}_i(t) = (g)\tilde{d}_i(t) + (1 - g)\tilde{w}_i(t)$$

Let D_i be the corresponding average end-to-end delay experienced by the AF classes AF_i. Let the delay differentiation ratios be $\Delta_i = D_i/D_{i+1}$. The overall average delay D_i is inversely related to the normalized delay $\tilde{h}_i(t)$, because the scheduler serves the queue with the highest priority or maximum normalized delay. In other words, if $\tilde{h}_i(t)$ gets higher, then that particular queue is more likely to be served and the corresponding overall delay D_i will be reduced. Therefore, the instantaneous normalized delay $\tilde{h}_i(t)$ reflects the average long term delay D_i. Hence, the delay differentiation ratios can then be represented as follows:

$$\Delta_i = \frac{D_i}{D_{i+1}} \approx \frac{\tilde{h}_{i+1}(t)}{\tilde{h}_i(t)}, \tag{3.1}$$

The new scheduler works to maintain the delay differentiation ratios Δ_i at a desired level by varying the AF class weights q_i, (i = 0, 1, 2...).

Whenever a packet is served, Δ_i is computed using Equation (3.1). Ideally, if the scheduler delay differentiates perfectly, Δ_i will always be equal to a desired (constant) value K. But this does not happen always. So in our scheme, when the delay ratio is greater (less) than K, the weights are adjusted, so that the delay ratio becomes equal to K. The experiments performed with this setting showed an increase in computation, since weights were recalculated upon the arrival of every packet. In order to avoid this computational overhead, we relaxed the condition as follows. If the delay ratio falls inside a window ($K - \epsilon$, $K + \epsilon$) around the desired value K, the scheduler parameters are left unchanged. Here, ϵ is the window width. The scheduler adjusts the weights whenever the delay differentiation ratio Δ_i deviates from its corresponding window ($K - \epsilon$, $K + \epsilon$).

In order to maintain the delay differentiation ratio between classes, the weights are changed according to the weight function:

$$f(q_i) = \begin{cases} q_i = q_i + \Psi \quad \& \quad q_{i-1} = q_{i-1} - \Psi & for \quad K < 2 - \epsilon \\ q_i = q_i^{init} \quad \& \quad q_{i-1} = q_{i-1}^{init} & for \quad K - \epsilon < \Delta_i < K + \epsilon \\ q_i = q_i - \Phi \quad \& \quad q_{i-1} = q_{i-1} + \Phi & for \quad \Delta_i > K + \epsilon \end{cases} \tag{3.2}$$

where q_i^{init} is the initial value of weight *i*. This function was formulated based on the property mentioned in [5], which states that decreasing/increasing the weight of a class affects the average delay of all other classes as well as its own average delay. In the experiments that were performed, ϵ was set to 0.25. ϵ is set based on a tradeoff between the number of computations and the stringent maintenance of the delay ratio. Setting ϵ = 0 would result in weight update computations upon every packet arrival, while a very higher value of ϵ (say $\epsilon > 1$), would result in a performance similar to the original HPD scheme. In (3.2), Ψ and Φ are calculated as:

$$\Psi = \frac{(q^i_{max} - q^i_{curr}) \times |(K - \epsilon) - \Delta_i|}{(q^i_{max} - q^i_{min})}$$

$$\Phi = \frac{(q^i_{max} - q^i_{curr}) \times |\Delta_i - (K + \epsilon)|}{(q^i_{max} - q^i_{min})}$$

where q^i_{max} and q^i_{min} are the maximum and minimum possible values of a class weight respectively, and q^i_{curr} is the current value of the weight in a cycle. $|\Delta_i - (K \pm \epsilon)|$ represents the deviation of the computed delay differentiation ratio Δ_i from the window ($K - \epsilon, K + \epsilon$). The difference $(q^i_{max} - q^i_{min})$, is the range of weights and $(q^i_{max} - q^i_{curr})$ is the maximum value by which the current value of a weight can be increased or decreased. Thus, deviations in the delay ratios are corrected by an increase or decrease of the weights in a linear fashion.

The class weights initially take the values of the inverse of the corresponding δ's, the delay differentiation parameters. For example, when the initial ideal weight values are set as q_1 = 1, q_2 = 2, and q_3 = 4, the desired delay ratio, which (in ideal cases) must be proportional to the ratio of weights, is Δ_i = q_{i+1}/q_i = K = 2 . Table 3.1 gives the corresponding maximum and minimum weights for each of the AF classes. The maximum (minimum) value of a particular weight is computed as the average of the weight's initial value and the initial value of the next higher (lower) weight.

Table 3.1 Maximum and minimum weights for each class.

Class	Initial Weight	Maximum Weight	Minimum Weight
AF1	q^0_{init} = 1	q^0_{max} = 1.5	q^0_{min} = 0.5
AF2	q^1_{init} = 2	q^1_{max} = 3	q^1_{min} = 1.5
AF3	q^2_{init} = 4	q^2_{max} = 6	q^2_{min} = 3

The action taken by the proposed AHPD scheduler when each and every packet arrives is described by the followings:

1. Initialization. Set the initial parameters q_i, g, and the desired delay differentiation ratio Δ_i. Compute initial $\tilde{h}_i$. When no packet has been served, select the queue to start service using the initial ideal weights q_i.

2. Whenever a queue is served, update the parameters as follows.

   (a) Calculate new Δ_2 using equation (3.1).

   (b) Update the weights q_3 and q_2 using equation (3.2).

   (c) Calculate new Δ_1 using equation (3.1).

   (d) Update weight q_1 using equation (3.2).

   (e) Compute the new normalized average delay by using equation (2.4).

3. Select the queue with the maximum normalized average delay and serve that queue.

4. Save the updated weights for the next cycle.

5. Go back to 2.

To summarize, we first update the weights of the two highest priority classes. Then the weight of the next low priority class is updated based on the weight of its predecessor.

3.2 Proportional Bandwidth Mechanism

The basis for a relative bandwidth differentiation is that a combination of the packet delay and loss of a flow reflects on the overall throughput achieved by the flow [25]. In a multi-class RED environment, the traffic classes are treated with different sets of RED parameters. The three RED parameters, namely, minimum threshold (min_{th}), maximum threshold (max_{th}), and the maximum drop probability (max_p), govern the packet accept/discard behavior at network nodes [14]. Dovrolis in [6] stated that the multi-class RED scheme is not suitable for a proportional loss differentiation. We do not contradict this statement, but propose to make use of multi-class RED to achieve a proportional bandwidth differentiation. While the adaptive HPD scheduler described above controls the packet delay, the class based dropper takes care of packet loss rate. These two actions reflects on the bandwidth or throughput achieved by the traffic class.

3.2.1 Colored RED

As mentioned earlier, in [25] a weighted RED (WRED) scheme was use to achieve a relative bandwidth differentiation between individual TCP flows. In WRED, more than one set of RED parameters are used to bandwidth differentiate flows. Motivated by this, we propose the use of a RIO-like[1] packet marking and dropping scheme, where the RED drop probabilities of the different classes are proportional to each other. The major difference

[1]RIO is RED with In/Out packets. It has two classes which differ in RED drop thresholds [4]

is that we achieve a per-class differentiation instead of per-flow differentiation. The RED parameters are fixed according to the following relation:

$$\frac{RED\ drop\, probability\ of\ class\ i}{RED\ drop\, probability\ of\ class\ j} = \frac{\sigma_i}{\sigma_j} \tag{3.3}$$

where σ is the loss differentiation parameter. As mentioned earlier, according to the AF PHB definition [13], there are currently 4 AF classes and within each of the AF classes, packets may belong to one of the three drop precedences. In the simulation experiments, we applied our scheme on a system with 3 AF classes, each with 2 drop precedences.

In our scheme, the sender marks the packets with the appropriate DSCP. When an edge router receives a packet, it checks the packet's DSCP and then marks it as either green, yellow or red colored packet. The RED parameters differ proportionally for the three colors. All the flows within a class are marked with the same color. The packets with different colors experience different accept/discard treatment. We call our scheme colored-RED (CRED), since the RED parameters are different for different colors. Unlike other similar multi-class RED schemes which treat individual flows, our CRED scheme acts on the traffic classes (aggregates).

The ability of CRED scheme to bandwidth differentiate greatly depends on the way its RED parameters (max_{th}, min_{th}, and max_p) are set for the different colors. In CRED, the colors differ in max_p values alone. The other parameters are the same for all classes or colors. Another factor that affects the CRED's performance is the calculation of average queue size. The average queue size value determines whether a particular packet is within the drop threshold limits. Here, we describe the way the max_p values are set for different colors, followed by CRED's average queue size computation.

3.2.2 Setting CRED Maximum Drop Probabilities

In CRED, the packets belonging to AF3, AF2, and AF1 classes are colored green, yellow and red respectively. The drop probability for AF3 class is the smallest of the three. The average queue size and the maximum drop probability are calculated in such a way that the condition given by (2.5) is met. Equation (2.5) which defines proportional bandwidth differentiation, is repeated here for convenience. It is given as:

$$\frac{BW_i}{BW_j} = \frac{\beta_i}{\beta_j}$$

The maximum drop probability max_p of the different AF classes is fixed in the same way as in [25]. The max_p values of the different colors(classes) are related as: max_p(red) = QDP * max_p(yellow) and max_p(yellow) = QDP * max_p(green), where QDP is the quality differentiation parameter. In this case the quality metric is bandwidth. Fig. 3.2 gives the drop probability versus the average queue size for three color WRED. It is clear from the figure that the colors differ only in max_p values and not in max_{th} or min_{th}.

3.2.3 Average Queue Size Calculation

The average queue size (AQS) of the AF_i (i = 1, 2, 3) class is computed based on the number of the AF_i class packets alone. A major difference between CRED and the scheme

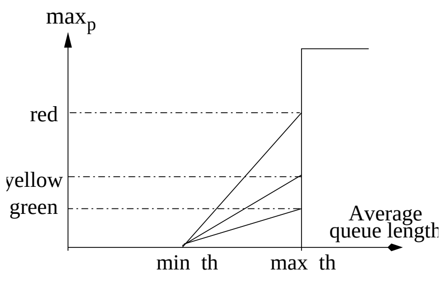

Figure 3.2 WRED Drop Probabilities

used in [25] is that, in CRED, the AQS calculation of a class is independent of the other class packets. In [25] the average queue size is computed as:

AQS_{AFi} = TSW estimate based on all (AF1 + AF2 + AF3) packets.

where i = 1, 2, 3.

But in CRED,

AQS_{AFi} = TSW estimate based on the AFi packets alone.

This is done in order to adhere to the AF PHB specifications [13], which state that the servicing of one AF class must be independent of the other AF classes. The AQS computation in CRED is also similar to the AQS calculation in the *decoupled* version of RIO [4], where the AQS of *In* and *Out* packets are independently determined.

CHAPTER IV

PERFORMANCE EVALUATION

The performance of the proposed combination of the adaptive HPD and colored RED schemes is evaluated through simulation experiments. All experiments are performed using the network simulator ns-2 [28]. Several traffic types are used to evaluate the robustness of the proposed scheme. They are as follows:

1. FTP sources.
2. Constant bit rate (CBR) sources.
3. On-/Off- sources with burst and idle times exponentially distributed.
4. On-/Off- sources with burst and idle times Pareto distributed.
5. Sources differing in RTT.

Notable sources are the On-/Off- traffic sources with burst and idle times taken from the Pareto and exponential distributions. The average value of the burst and idle times are set to 500 msec each. TCP and UDP flows are used in some of the experiments to study their interactions. All the TCP agents use the selective acknowledgment (SACK) mechanism [21]. A fixed packet size of 1000 bytes is used in all the experiments. The value of 'g' in (2.4) is set to 0.85.

The proposed scheme is compared with a combination of the original HPD packet scheduler [5] and the RIO dropping scheme[4]. For this combination, the same RIO parameters are used for all the classes: (10, 20, 0.04) for OUT[1] packets and (20, 40, 0.04) for IN[2] packets, where the three parameters represent ($min_{th}, max_{th}, max_p$), respectively.

4.1 Simulation Model

Fig. 4.1 depicts the topology used in all the simulation experiments. The topology consists

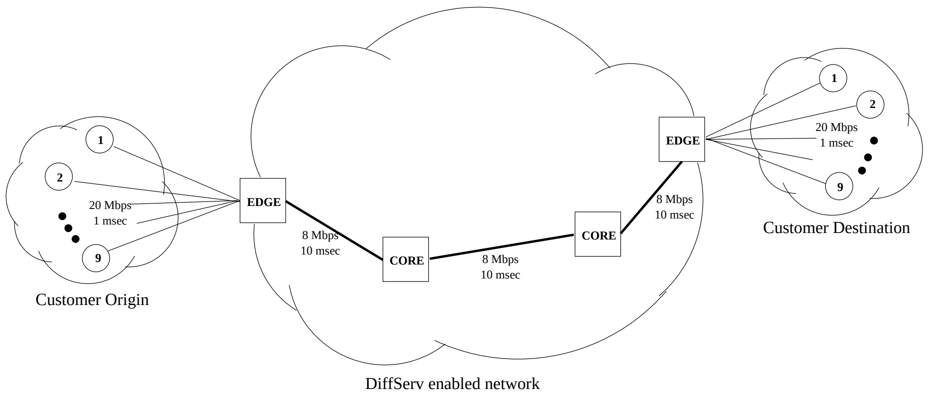

Figure 4.1 Simulation Topology

of 9 sources and 9 destinations connected through 2 edge routers and 2 core routers. The edge routers differ from the core routers in the way that, they have built-in packet meters, policers, and markers. The core routers on the other hand do not meter or mark the packets. They only employ the proposed adaptive HPD scheduler and CRED dropper. The packet meters use a time sliding window technique (TSW) [4] to compute each flow's instantaneous sending rate. This instantaneous sending rate is used to determine the packet's

[1]Packets that fall outside service agreement

[2]Packets that fall inside service agreement

Table 4.1 Parameters for CRED

Queue Type	min_{th}	max_{th}	max_p
Red / AF11 (IN)	20	40	0.08
Red / AF12 (OUT)	10	20	0.16
Yellow / AF21 (IN)	20	40	0.04
Yellow / AF22 (OUT)	10	20	0.08
Green / AF31 (IN)	20	40	0.04
Green / AF32 (OUT)	10	20	0.02

drop probability. The packets from the customer network are classified to one of the 3 AF classes based on the agreement between the service provider and the customer, and then the packets are marked as green, yellow or red accordingly. Further, the edge routers meter the flows and subject the packets to one of the two drop threshold levels. Table 4.1 gives an example of a parameter set used in the simulation experiments. In all the routers the buffer length is set high enough so that the queues never experience buffer overflows. In all the cases, the bottleneck link bandwidth is set to 8 Mbps. All the simulation experiments are performed for a period of 300 seconds. The flows are preallocated bandwidth according to a service level agreement. Experiments were carried out using different traffic types so as to validate our claim that our scheme provides a proportional bandwidth, delay and loss differentiation under several network conditions.

4.2 Simulation Experiment Results

In order to analyze the performance of our scheme, we made the following measurements:

1. The average throughput achieved by each flow. The throughput is calculated as the sum of the bytes successfully transmitted divided by the total simulation time.

2. The average one way end-to-end delay of each flow. This is the average of the delay experienced by every packet of the flow.

3. The packet loss rate. This is the ratio of the number of packets lost to the total number of packets transmitted.

The above three measurements are presented in the form of bar plots. The three horizontal lines in the throughput differentiation bar plot represent the target rates of the three AF classes: 380 Kbps, 760 Kbps and 1520 Kbps respectively. Likewise, the horizontal line in the delay differentiation bar plots represents the minimum one way end-to-end delay that each packet would experience. This delay is 32 msec in all the simulation cases except in the last test case where all the flows differ in the round trip times (RTT). Assuming that the packet transmission delays, the packet processing delays, and the queuing delays are negligible, the 32 msec then consists only of the propagation delays alone.

Apart from the above three measurements, the following were also calculated for the analysis:

1. Delay ratio between classes. The class delay is computed as the average of the delay experienced by the flows constituting the class. From this the delay ratio between classes is computed.

2. In some experiments, the bandwidth ratio is also calculated in a fashion similar to that of delay ratio.

3. Fairness index. The Fairness index was calculated using the formula given in Appendix A.

4. In the experiment that used flows differing in RTT, the percentage of increase in average delay from the ideal value is calculated.

4.2.1 Experiments with FTP traffic over TCP SACK

In the first set of experiments, FTP traffic from greedy sources is carried over TCP by all the nine flows. Fig. 4.2 shows the bandwidth, delay, and loss differentiation achieved by our scheme. Fig. 4.3 shows the differentiation achieved using the original HPD scheme. There is not much difference in the bandwidth differentiation between our scheme and the original HPD. But in the case of a delay differentiation, our scheme maintains the required delay differentiation ratios better than the original HPD scheme. In our scheme, the delay differentiation ratios Δ_1 and Δ_2 are very close to 2 (the desired value). This is achieved by the reduction in the delay levels of AF3 class and an increase in the delay levels of AF1 class. Table 4.2 shows the average delays and the delay ratios for each of

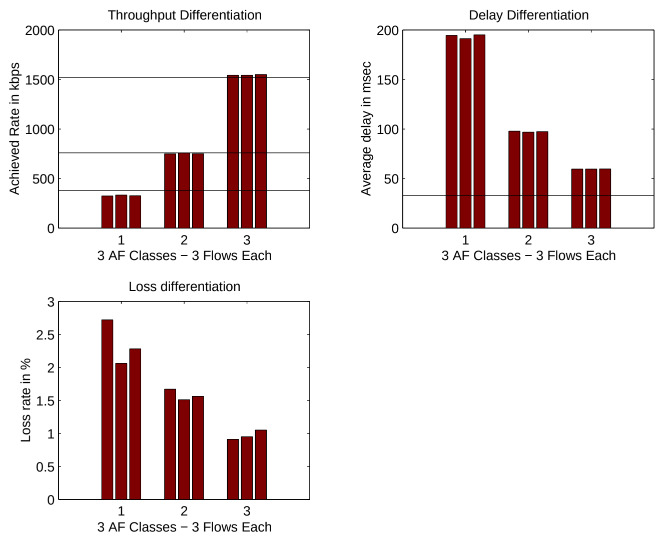

Figure 4.2 AHPD and CRED: FTP over TCP SACK.

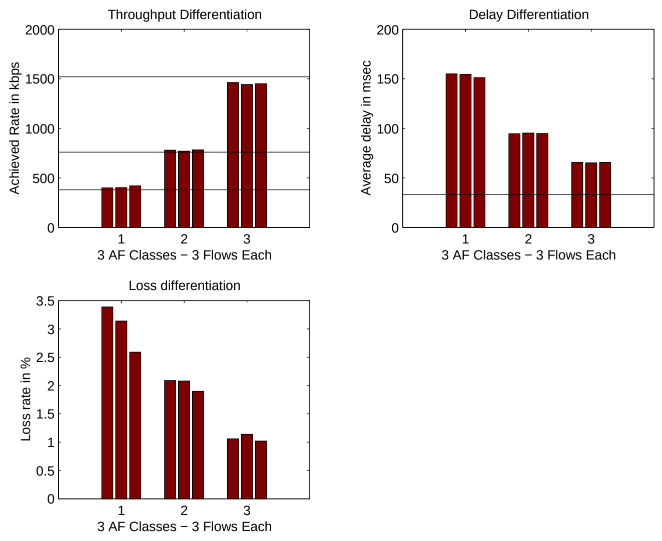

Figure 4.3 Original HPD and RIO: FTP over TCP SACK.

the schemes. It is clear that our proposed scheme works better in maintaining the delay ratios between classes. It should also be noted that our scheme achieves lower loss rates

Table 4.2 Delay Comparison for FTP Sources

Scheme	Average Class Delay msec			Delay Ratio	
	AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
AHPD & CRED	193.67	97.4	59.66	1.99	1.63
HPD & RIO	153.6	94.94	65.53	1.62	1.45

than the original HPD scheme. Hence, traffic is differentiated well in terms of all three performance metrics.

4.2.2 Experiments with Constant Bit Rate Sources

Three different experiments are performed with CBR sources. In the first case, all the flows carry CBR traffic over TCP. In the second case, one flow in each AF class carries CBR traffic over UDP while the other flows carry FTP traffic over TCP. In the last experiment, all the flows consist of CBR traffic over UDP. The corresponding bandwidth, delay, and loss differentiation for the 3 simulation experiments are shown in Fig. 4.4, 4.5 and 4.6. In all the cases, the CBR sources generate packets at a rate greater than their target rates. When such a CBR sources transmits over UDP, it greatly misbehaves and does not adhere to service agreements.

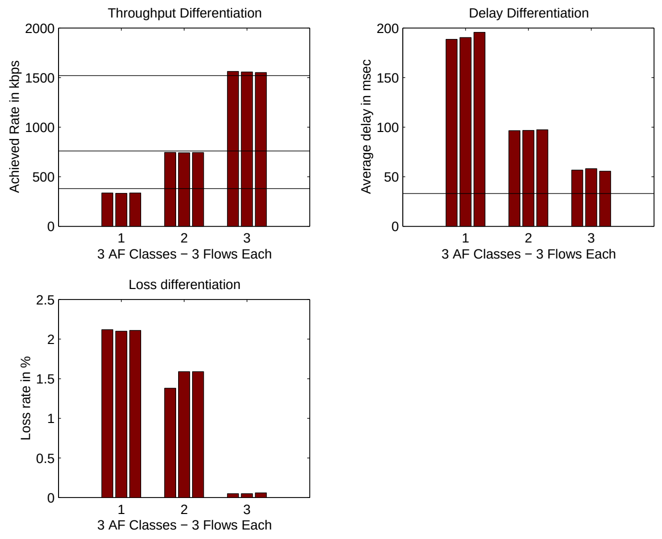

Figure 4.4 AHPD and CRED: All 9 flows are CBR over SACK

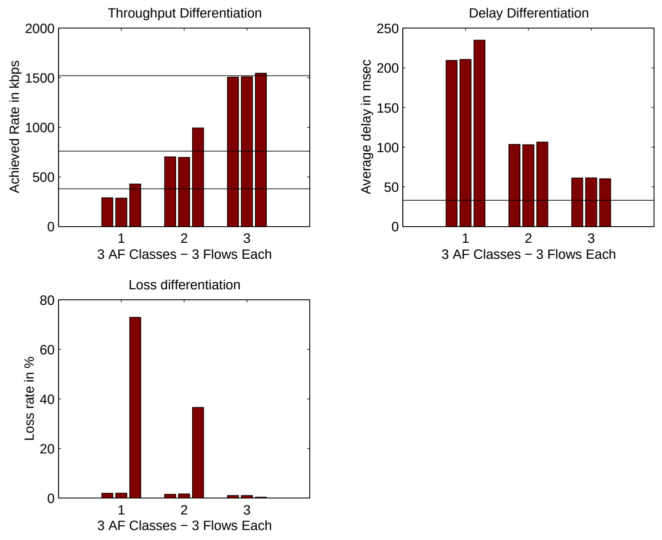

Figure 4.5 AHPD and CRED: One flow in each class is CBR/UDP. Rest FTP/SACK

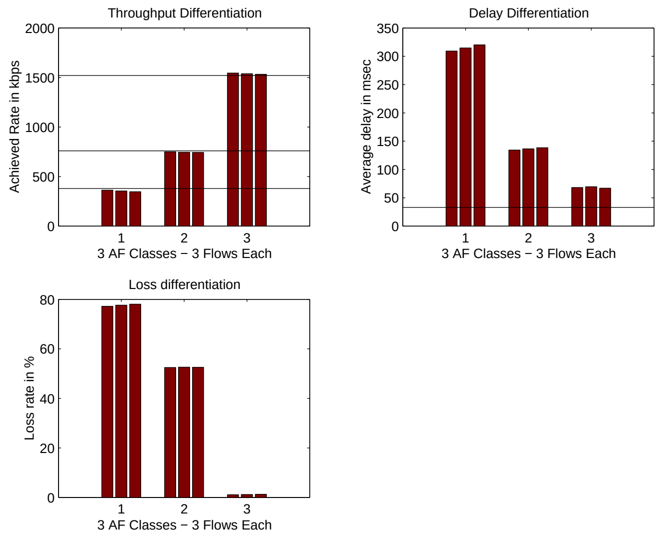

Figure 4.6 AHPD and CRED: All 9 flows are CBR/UDP

We see that our proposed scheme achieves the desired bandwidth, delay and loss differentiation in the first and third experiment. Bandwidth and delay differentiation is also attained in the second case, but the TCP/UDP interaction effect arises in this case. From Fig. 4.5, the UDP flows within each class appears to get a higher share of bandwidth than TCP. The fairness index between flows within each class in the second experiment was calculated as 0.96, 0.97, 0.99 for AF1, AF2 and AF3 classes respectively. This level of fairness is quite acceptable for practical purposes.

A very important fact to note is that, the non-responsive UDP flows are heavily punished when they try to exceed their allocated share of the bandwidth. However, these UDP flows still attain their target rates. All the packets lost by the UDP flows were in excess of their service level agreements. The severity of the UDP packet drop also depends on the packet's class. Hence, we can observe from Fig. 4.5 that a loss differentiation is also achieved. Since the TCP sources regulate themselves during congestion, they experience a far lower packet loss than the UDP sources. Table 4.3 shows the delay comparison between the proposed scheme and the original HPD scheme. As in the earlier case with FTP,

Table 4.3 Delay Comparison for CBR Sources

Traffic Type	Scheme	Average Class Delay msec			Delay Ratio	
		AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
All flows are CBR/TCP	AHPD & CRED	191.65	96.90	56.83	1.97	1.7
	HPD & RIO	152.55	94.47	65.36	1.61	1.45
One flow in each class is CBR/UDP	AHPD & CRED	218.33	104.43	60.92	2.09	1.71
	HPD & RIO	170.48	103.24	69.67	1.65	1.48
All flows are CBR/UDP	AHPD & CRED	314.78	136.53	68.29	2.3	1.99
	HPD & RIO	216.49	126.32	81.13	1.71	1.56

delay differentiation ratio is better maintained using our scheme than the original HPD scheme. Thus, bandwidth, delay, and loss differentiation is achieved simultaneously.

Table 4.4 Delay Comparison for EXP Sources

Traffic Type	Scheme	Average Class Delay msec			Delay Ratio	
		AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
Exponential over TCP	AHPD & CRED	186.64	94.94	58.66	1.97	1.61
	HPD & RIO	151.83	94.07	65.10	1.61	1.45
Exponential over UDP	AHPD & CRED	241.61	123.46	70.49	1.96	1.75
	HPD & RIO	217.43	128.93	82.38	1.68	1.56

4.2.3 Experiments with Exponential Traffic Sources

Two different simulation experiments were performed with the On-/Off- traffic sources whose burst and idle times are exponentially distributed. In the first case, all the flows carry traffic from Exponential source over TCP and in the other case, all traffic is carried over UDP. The corresponding results are shown in Fig. 4.7 and 4.8. In both the cases, a bandwidth, delay and loss differentiation is achieved simultaneously. As in the earlier cases, the new scheme maintains the delay ratio better than the original HPD scheme. The delay comparison is shown in Table 4.4.

In the experiment with traffic over UDP, the UDP flows experience losses greater than 50%. Thus, the dropping scheme effectively punishes AF1 and AF2 classes when they

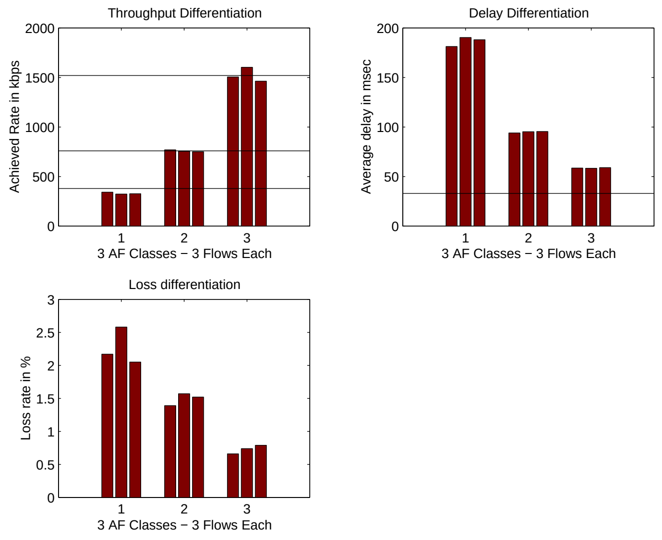

Figure 4.7 AHPD and CRED: Exponential traffic over SACK

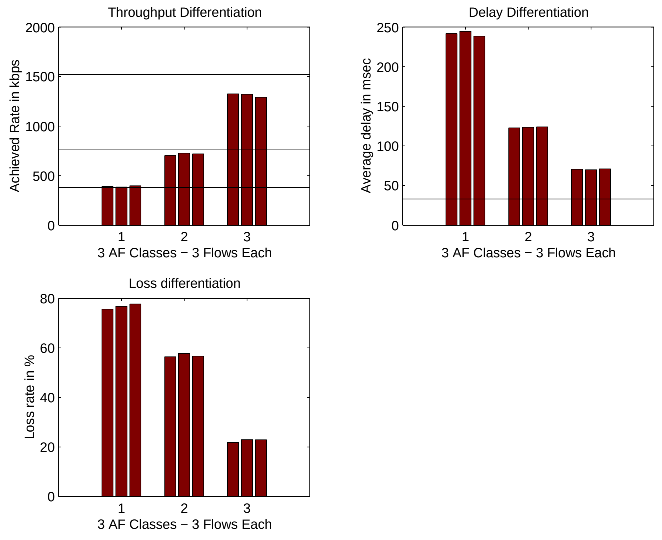

Figure 4.8 AHPD and CRED: Exponential traffic over UDP

try to obtain a greater share of the link capacity. Again, our scheme maintains the delay differentiation ratio better than the original scheme.

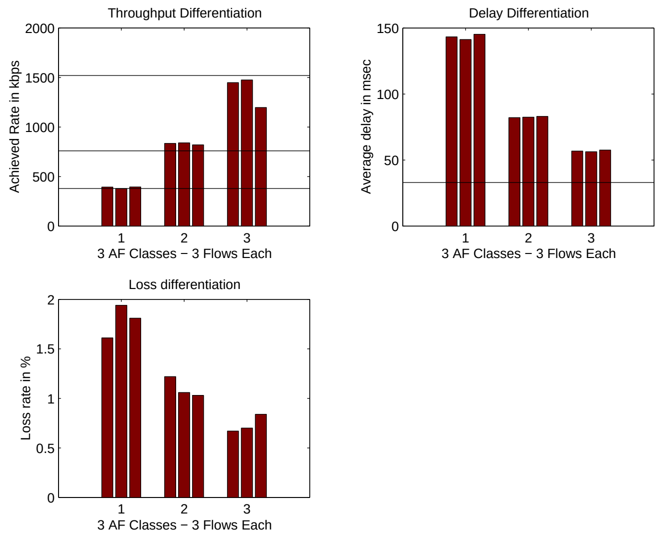

Figure 4.9 AHPD and CRED: Pareto over SACK

4.2.4 Experiments with Pareto Traffic Sources

Two more different simulation experiments were performed with On-/Off- sources with burst and idle time following the Pareto distribution. The distribution's shape parameter α is set to 1.2. Traffic sources adhering to Pareto arrival pattern, represent the bursty nature of today's Internet. In the first experiment, packets from Pareto sources are carried over TCP and in the other, all traffic is carried over UDP. The corresponding results are shown in Fig. 4.9 and 4.10. As it can be observed from the figures and the delay comparison Table 4.5, a bandwidth, delay and loss differentiation is achieved for both of the cases. The delay differentiation ratios are also maintained better than the original HPD scheme.

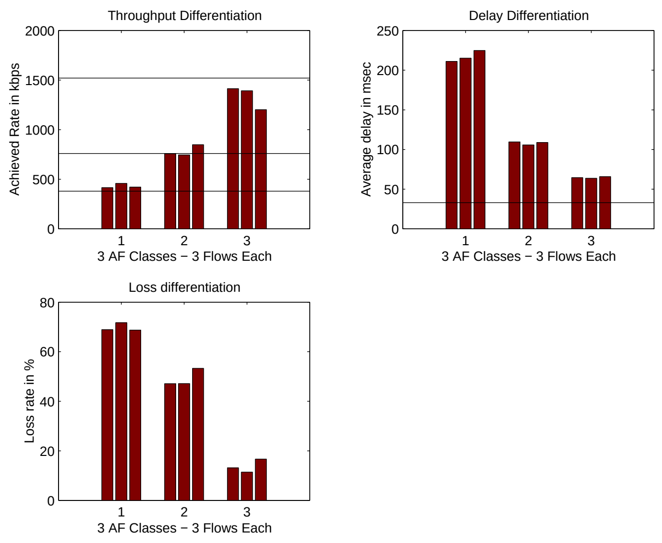

Figure 4.10 AHPD and CRED: Pareto over UDP

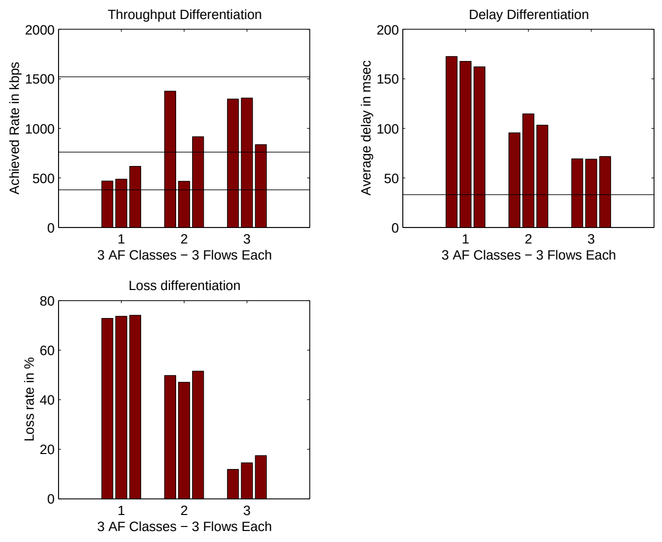

Figure 4.11 Original HPD and RIO: Pareto over UDP

In the first experiment with Pareto traffic over TCP (Fig. 4.9), the bandwidth distribution appears to suffer. Although, one of the AF3 flows in Fig. 4.9 appears to receive a

Table 4.5 Delay Comparison for Pareto Sources

Traffic Type	Scheme	Average Class Delay msec			Delay Ratio	
		AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
Pareto over TCP	AHPD & CRED	143.33	82.59	56.93	1.73	1.45
	HPD & RIO	135.96	86.39	61.40	1.57	1.41
Pareto over UDP	AHPD & CRED	181.27	97.22	63.12	1.86	1.54
	HPD & RIO	167.46	104.46	69.96	1.60	1.49

less share of the bandwidth, the fairness index calculated proves otherwise. The fairness indices calculated for the 3 classes when Pareto traffic was carried over TCP SACK are 0.99, 0.99 and 0.99 for AF1, AF2 and AF3 classes respectively. The delay differentiation ratios are also strictly maintained by our scheme, which is not the case with the original HPD scheme.

Remarkably, Fig. 4.11 shows the failure of the original HPD and RIO combination to bandwidth differentiate when the bursty traffic is carried over UDP. On the other hand, our scheme distributes the bandwidth in the desired manner and the delay differentiation ratios are properly maintained. Proportional loss differentiation is also achieved in both the experiments. With the increasing number of applications using UDP, the ability of

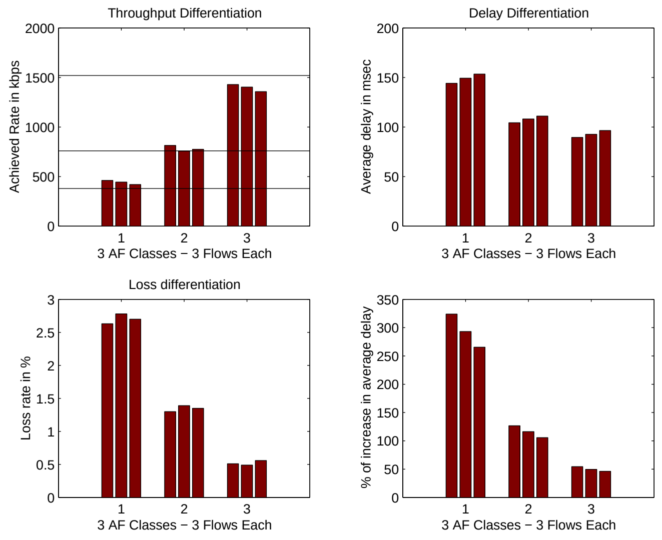

Figure 4.12 Original HPD and RIO: Flows with Different RTT

our scheme to effectively bandwidth, delay, and loss differentiate bursty traffic over UDP supports our claim that our scheme is more robust.

4.2.5 Experiments with Flows Differing in RTT

The performance of the proposed scheme is tested with flows having different round trip times (RTT). The 9 flows starting from class AF1 to AF3 have RTTs of 68, 76, 84, 92, 100, 108, 116, 124, and 132 msecs respectively. Thus, a higher priority flow has a higher RTT than a lower priority flow. Fig. 4.12 and 4.13 show the performance of the original HPD scheme and our proposed scheme respectively.

The proposed CRED scheme suppresses the effect of difference in RTT between the flows and thus the classes get a proportional share of bandwidth. On the other hand, the bandwidth ratios between classes is affected in the experiments with the original scheme.

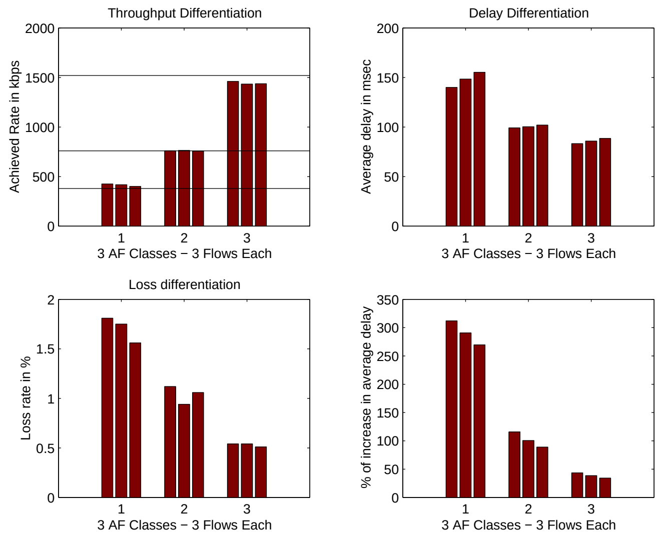

Figure 4.13 AHPD and CRED: Flows with Different RTT

This is because the low priority classes, due to their lower RTTs, get a greater than allocated share of bandwidth. Table 4.6 shows the average bandwidth obtained by the different classes and the bandwidth ratios between the AF classes for the two test cases. Clearly, our proposed scheme achieves a proportional bandwidth differentiation better than the original HPD and RIO combination.

Regardless of the low priority classes having lower RTTs, both the original HPD and our scheme succeed in achieving a delay differentiation. Table 4.7 shows the delay ratio comparison between our scheme and the original HPD scheme. The difference in the RTT has however affected the delay ratios between the classes. Again, our scheme shows an improved performance over the original HPD scheme. Fig. 4.12 and 4.13 also show the percentage of the increase in the delay of the individual flows from their ideal one way delay values. Table 4.8 gives the corresponding numerical values. In most of the cases our

Table 4.6 Bandwidth Comparison for RTT Experiment

Scheme	Average Class Bandwidth kbps			Bandwidth Ratio	
	AF1	AF2	AF3	$\frac{AF2}{AF1}$	$\frac{AF3}{AF2}$
AHPD & CRED	415.12	761.11	1433.90	1.83	1.89
HPD & RIO	442.43	782.98	1397.22	1.77	1.78

scheme increases the end-to-end delay to a lesser extent than the original scheme. Hence, our scheme achieves proportional delay, loss and bandwidth differentiation simultaneously and also shows great resilience to variation in RTT.

Table 4.7 Delay Comparison for RTT Experiment

Scheme	Average Class Delay msec			Delay Ratio	
	AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
AHPD & CRED	147.97	100.50	85.90	1.47	1.17
HPD & RIO	149.02	107.87	92.93	1.38	1.16

Table 4.8 Percentage of Delay increase from ideal values.

Adaptive HPD (in %)	Original HPD (in %)
312.00	323.97
290.71	293.11
269.69	265.55
115.80	126.80
100.62	116.32
88.93	105.79
43.53	54.44
38.48	49.52
34.26	46.22

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, we presented a novel scheduling technique (Adaptive HPD) to proportionally delay differentiate service quality. The adaptive HPD scheduler fine tunes its weight by monitoring the delay difference between the traffic classes. The simple, scalable, and robust scheduler maintains the delay ratio between traffic classes under various network conditions. In addition, we used a class based dropper (CRED) to achieve proportional loss differentiation. The combination of the scheduler and the dropper also results in a proportional bandwidth differentiation. Thus, we simultaneously attain a proportional bandwidth, delay, and loss differentiation.

The resource demanding and heterogenous nature of today's Internet applications cannot be handled appropriately by the simple best-effort Internet architecture. The best-effort Internet model will soon be obsolete. Future applications will demand service guarantees from the service provider. The Internet user will no longer be satisfied with a mere promise of *good* service. On the other hand, the service providers will begin to see service differentiation as a means of revenue. By offering services which differ in quality, more customers can be attracted. QoS provisioning methods not only give the ISPs a competitive edge, but also are a source of revenue. Thus, the ISPs will greatly make use of QoS provisioning methods to satisfy all these customer needs. Our proposed scheme is a very good solution to all these demands. The significance of our scheme lies in the fact that it is simple, unified, robust and above all, controls all three quality metrics. Thus, while applications are offered services proportionally differing in terms of delay and packet loss, the ISP can make use of our scheme to distribute bandwidth between classes proportionately.

We tested our scheme with various traffic sources. The results obtained from these simulations experiments suggest the following:

1. The need for maintaining the delay ratio between classes grows as the number of applications increases. When service policies are provisioned, care must be taken to assign delay tolerant applications to the low priority classes. Experimental results prove that the delay differentiation ratios obtained using our scheme is very much close to the ideal desired value. Moreover, our scheme also maintains the delay ratios better than the original HPD scheme.

2. Setting the parameters for the RED portion of the CRED scheme is not trivial. A better bandwidth differentiation ratio can be achieved by using stricter RED parameter values and thus punishing the misbehaving flows in a harsher manner. Queuing delays at the routers and packet losses must also be taken into consideration while setting RED parameters.

3. Loss rates and loss differentiation is better in the experiments which use congestion responsive transport agents (TCP) than those experiments which use non-responsive transport agents (UDP). A good example of this case is the experiment with the CBR sources. Nevertheless, the non-responsive flows are also proportionally loss differentiated, but have to pay a price in the form of a higher loss rate. Also, CBR traffic, especially over UDP greatly affects the fair distribution of resources in an uncontrolled environment. The UDP flows with AF1 and AF2 class labels, experience very high packet loss rates, since these flows try to take advantage of the TCP flows and obtain a more than allocated resource share. Our scheme effectively punishes the misbehaving UDP flows appropriately and distributes bandwidth proportionally. Since the scheduler works entirely based on the packet's class rather than the underlying transport mechanism (TCP/UDP), the delay differentiation is not affected by the presence of UDP. Hence, our scheme is robust even in the presence of misbehaving UDP flows.

4. The experiment with Pareto sources further augments our claim that our scheme is more robust. Our scheme out-performed the existing HPD and RIO combination in these tests with bursty traffic. For example, when bursty traffic was carried over UDP, the existing algorithm (HPD and RIO) even failed to bandwidth differentiate, while our scheme bandwidth differentiated in a far better manner. Hence, our scheme is more robust in achieving a bandwidth, delay, and loss differentiation simultaneously.

5. Networks have a bias against flows with longer RTTs. The ability of the dropping mechanism to deliver bandwidth in a proportional manner in-spite of this bias, proves the tolerance of the dropping scheme to RTT effects.

With the support of the simulation experiment results, we argue that the bandwidth, delay, and loss can be controlled simultaneously by acting on the delay and packet loss alone. We further compare our scheme with the popular combination of HPD and RIO scheme. We conclude that our proposed combination of an adaptive scheduler and a packet dropper is more robust than the combination of original HPD and RIO.

5.2 Future Work

This thesis work gave birth to new ideas along the same directions. These ideas if successful would contribute greatly to proportional differentiated services model.

The original HPD scheduler was designed with the purpose of making a balance between the WTP scheduler and the PAD scheduler. In the HPD scheme, the factor 'g' decided the balance between WTP and PAD. Although experiments with a fixed 'g' resulted in good results, the scheduling scheme can be made more robust by making this factor adaptive to packet arrival pattern. Hence, as the arrival pattern of packets varies over a small window of time, the factor 'g' can also be varied accordingly.

The class based packet dropper, CRED, drops packets based on the TSW metering of the flows. The packet dropper can be made more robust by maintaining a history of the packet loss rate. The count of packets lost by every class can be compared and their respective drop probabilities can be modified accordingly. This would help maintain the proportional loss differentiation in a robust manner.

A much broader topic to work on from this stage would be to test the effectiveness of the proposed DiffServ techniques in a Multiprotocol Label Switching (MPLS) enabled network[11][23]. Several recent works have contributed greatly to the success of a DiffServ architecture over laid on a MPLS network. The proposed scheme can be further fine tuned by making use of MPLS's traffic engineering mechanisms.

REFERENCES

[1] Y. Bernet, “The Complementary Roles of RSVP and Differentiated Services in the Full-Service QoS Network,” *IEEE Communications Magazine*, vol. 38, no. 2, February 2000, pp. 154–162.

[2] B. E. Carpenter and D. D. Kandlur, “Diversifying Internet Delivery,” *IEEE Spectrum*, vol. 36, no. 11, November 1999, pp. 57–61.

[3] Z. Chen, T. Yang, and D. Makrakis, “Realistic bursty traffic modeling for differentiated services network,” *Proceedings of IEEE Conference on PACRIM*, May 2001, pp. 575–578.

[4] D. D. Clark and W. Fang, “Explicit Allocation of Best-Effort Packet Delivery Service,” *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, August 1998, pp. 362–373.

[5] C. Dovrolis, *Proportional Differentiated Services for the Internet*, doctoral dissertation, University of Wisconsin - Madison, May 2000.

[6] C. Dovrolis and P. Ramanathan, “Proportional Differentiated Services, Part II: Loss Rate Differentiation and Packet Dropping,” *IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, June 2000, pp. 52–61.

[7] C. Dovrolis, D. Stiliadis, and P. Ramanathan, “Proportional Differentiated Services: Delay Differentiation and Packet Scheduling,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, February 2002, pp. 12–26.

[8] A. Dutta-Roy, “The Cost of Quality in Internet Style Networks,” *IEEE Spectrum*, vol. 37, no. 9, September 2000, pp. 57–62.

[9] L. Essafi, G. Bolch, and A. Andres, “An Adaptive Waiting Time Priority Scheduler for the Proportional Differentiation Model,” *Proceedings of ASTC HPC*, April 2001.

[10] L. Essafi, G. Bolch, and H. D. Meer, *Dynamic Priority Scheduling for Proportional Delay Differentiated Services*, Technical Report TR-I4-01-03, University of Erlangen-Nuremberg, Germany, [Online] Available: http://www4.informatik.uni-erlangen.de/TR/pdf/TR-I4-01-03.pdf 2001.

[11] F. L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, T. Cheval, and J. Heinanen, "MPLS Support of Differentiated Services," May 2002, Request for Comments 3270.

[12] P. Ferguson and G. Huston, "Quality of Service in the Internet: Fact, Fiction, or Compromise?," *INET, Geneva, Switzerland*, July 1998, pp. 21–24.

[13] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," June 1999, Request for Comments 2597.

[14] V. Jacobson and S. Floyd, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, August 1993, pp. 397–413.

[15] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," June 1999, Request for Comments 2598.

[16] R. Jain, *The Art of Computer Systems Performance Analysis*, vol. 1, John Wiley and Sons, 1991.

[17] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*, vol. 1, Wiley, 1976.

[18] M. K. H. Leung, J. C. S. Lui, and D. K. Y. Yau, "Adaptive Proportional-delay Differentiated Services: Characterization and Performance Evaluation," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, December 2001, pp. 801–817.

[19] C.-C. Li, S.-L. Tsao, M. C. Chen, Y. Sun, and Y.-M. Huang, "Proportional delay differentiation service based on weighted fair queuing," *Proceedings of The 9th International Conference Computer Communications and Networks*, May 2000, pp. 418–423.

[20] M. E. Markaki, M. P. Saltouros, and I. S. Venieris, "Proportional packet loss differentiation and buffer management for differentiated services in the Internet," *Proceedings of The 25th Annual IEEE Conference on Local Computer Networks*, June 2000, pp. 306–313.

[21] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," October 1996, Request for Comments 2018.

[22] L. Mathy, C. Edwards, and D. Hutchison, "The Internet: A Global Telecommunications Solution?," *IEEE Network*, vol. 14, no. 4, July/August 2000, pp. 46–57.

[23] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," January 2001, Request for Comments 3031.

[24] G. Ruzzo and N. Chiminelli, "WRED Tuning for Bottleneck Link," February 2000, Available: http://carmen.cselt.it/papers/wred-cern/home.html.

[25] T. Soetens, S. D. Cnodder, and O. Elloumi, "A relative bandwidth differentiated service for TCP micro-flows," *Proceedings of First IEEE/ACM International Symposium on Cluster Computing and the Grid*, August 2001, pp. 602–609.

[26] R. Stevens, *TCP/IP illustrated*, vol. 1, Addison Wesley, 1994.

[27] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, November 1997, pp. 8–18.

[28] UCB/LBNL/VINT, "The Network Simulator ns-2," May 2002, http://www.isi.edu/nsnam/ns/.

[29] X. Xiao and L. M. Ni, "Internet QoS: A Big Picture," *IEEE Network*, vol. 13, no. 2, March/April 1999, pp. 8–18.

APPENDIX A

FAIRNESS INDEX

The fairness index was calculated using the following formula as given in [16]:

$$Fairness\ \ Index\ \ FI = \frac{\left[\sum_i x_i\right]^2}{N \sum {x_i}^2}$$

where fairness index ranges between 0 and 1, x_i is the mean throughput of traffic source i, and N is the total number of sources under consideration. The closer the fairness index to 1, the fairer is the bandwidth distribution between sources.