Quality of Service Support in the Backbone Network of the Universal Mobile Telecommunications System Using DiffServ Model

by

Farshid Agharebparast

B.Sc. in Electrical Engineering, Isfahan University of Technology, Iran, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Department of Electrical and Computer Engineering)

We accept this thesis as conforming
to the required standard

The University of British Columbia

October 2001

Department of Electrical & Computer Engineering

The University of British Columbia
Vancouver, Canada

Date oct. 10, 2001

Abstract

The performance of the backbone bearer service of the Universal Mobile Telecommunications System (UMTS), or of its 2.5G precedent General Packet Radio Service (GPRS), has a considerable impact on the end-to-end Quality of Service (QoS) observed by a traffic flow. The purpose of this thesis is to provide a basic methodology for provisioning QoS in this backbone network by using Differentiated Services (DiffServ), which is an IP-based QoS technology. It focuses on two important issues. First, a basic plan for QoS mapping from one system to another is presented. Second, the structure of a DiffServ-based UMTS backbone router is studied which can provide the appropriate functionality of forwarding packets with PHBs equivalent to the original UMTS QoS.

In particular, three major components of a DiffServ router, including the scheduler, the dropper, and the meter, have been addressed and novel algorithms have been proposed to make them suitable for a UMTS backbone router. This thesis proposes an efficient, fair scheduling and link sharing system called DDB-FFQ, which has the capability of assigning decoupled delay bounds and bandwidth allocations. This paper also studies the implementation of Random Early Detection (RED) on a shared-memory buffer and proposes new schemes that have improved packet loss ratios. The performance of a multiple node backbone network is then evaluated by using the OPNET simulation package. The results show that in terms of delay and allocated bandwidth, the routers are able to provide the PHBs considered for each traffic aggregate. They also illustrate the link sharing capability of the router in times of congestion.

Contents

List of Tables

List of Figures

Acknowledgments

> *"There is no such thing as a 'self-made' man. We are made up of thousands of others. Everyone who has ever done a kind deed for us, or spoken one word of encouragement to us, has entered into the make-up of our character and of our thoughts, as well as our success."* *- George Matthew Adams*

I am appreciative of my advisor, Professor Victor C.M. Leung, for his support and valuable guidance during this project. He has been patiently available for any question or discussion anytime I needed him. It was a rewarding experience working with him.

During my course of study here at UBC, I enjoyed the friendship of a number of talented and amicable individuals. Specially, I would like to thank my colleagues in the Communications group: Cyril Iskander, Vaibhav Dinesh, Hansen Wang and Xinrong Wang. It was fun participating in technical and nontechnical discussions with them.

I am also grateful to the faculty and staff of the Electrical and Computer Engineering Department for their valuable lectures and assistance in making my study more enjoyable here. I also would like to express my appreciation to Ms. Kirsty Barclay for proofreading my thesis.

Finally, I would like to thank my parents for their deep love and unswerving devotion. This would never have happened without their dedication and encouragement.

This work was supported by Motorola Canada Ltd., Richmond, B.C., Canada and Natural Sciences and Engineering Research Council under Grant CRDPJ223095.

To my beloved parents.

Chapter 1

Introduction

1.1 Motivation

Third generation (3G) mobile telecommunications systems are promising technologies that will bring a variety of new services to mobile users. They are aimed at providing full access to multimedia, real-time and data applications, such as mobile Internet access, email service, video streaming and voice over IP, with a reasonable grade of service. To achieve these goals, these systems should be able to offer differentiated services to different traffic flows, depending on the demanded Quality of Service (QoS). The ideal of this QoS structure is to be "future proof", to be able to accommodate any potential new application. To provide a smooth evolution from a second generation (2G) wireless system to a 3G system, 2.5G architectures are proposed. A 2.5G system is usually an enhancement of a 2G system, making a higher rate of data services available.

The Universal Mobile Telecommunications System (UMTS) is a promising 3G wireless system based on the Global System for Mobile communications (GSM). GSM is a 2G wireless system used predominantly throughout the world. It was originally standardized by a European organization (ETSI), and is now the leading digital cellular technology, especially in Europe and Asia. General Packet Radio Service (GPRS) [1] is standardized by ETSI as a GSM enhancement providing true packet data services. It has the advantage of being built on the existing GSM network, and by only adding two new nodes to the GSM system and upgrading the software of other nodes, it can provide a true packet data system. GPRS also has more efficient radio link utilization and supports QoS. It is anticipated that GPRS's core network will evolve into the core network of UMTS. Due to the similarity between the QoS architecture and the backbone network structure of UMTS and GPRS release 99, the two terms are used interchangeably in most of the topics covered in this thesis, and therefore, any discussion of one can be applied to the other, unless otherwise stipulated.

Provisioning reasonable service to real-time applications is viable only when the system can guarantee a well-defined end-to-end service. The end-to-end QoS, that is, from one Terminal Equipment (TE) to another TE observed by a user, is actually the overall performance of different bearer services constructing an end-to-end connection path. The UMTS QoS specification [2] divides the end-to-end QoS into three main levels: the MT/TE bearer service, the UMTS bearer service, and the external bearer service. The MT/TE bearer service is the user's local interface, for example, between the user's laptop and his cellphone. QoS support of this bearer service is beyond the scope of the UMTS QoS specification. The external bearer service is the section of the datapath located outside the UMTS infrastructure, usually within the Internet. The UMTS bearer service itself consists of a radio interface and the core network. Figure 1.1 shows the GPRS/UMTS system infrastructure [3]. It also illustrates the path a packet might take when it is transmitted from a wireless TE to an Internet TE. Therefore, the overall QoS observed by this packet depends on the behavior of each network section it passes through.

The focus of this project is the study of the QoS support in the UMTS/GPRS core network. This backbone network is an IP network consisting of SGSN and GGSN nodes. Therefore, an IP-based QoS mechanism needs to be implemented to add service differentiation capability to this backbone network. A promising methodology for this purpose is the Differentiated Services (DiffServ). DiffServ has a number of favorable features, such as scalability, and a relatively simple implementation requirement. These characteristics make the DiffServ model a likely candidate for adding QoS support to the UMTS backbone.

IETF has standardized the basic configuration of a DiffServ router. A DiffServ router consists of a number of components connected together to perform a wide range of traffic management, shaping, and queuing functions. These elements can be configured in a number of different ways and there are numerous algorithms that can be implemented on each of them, depending on the functionality requested.

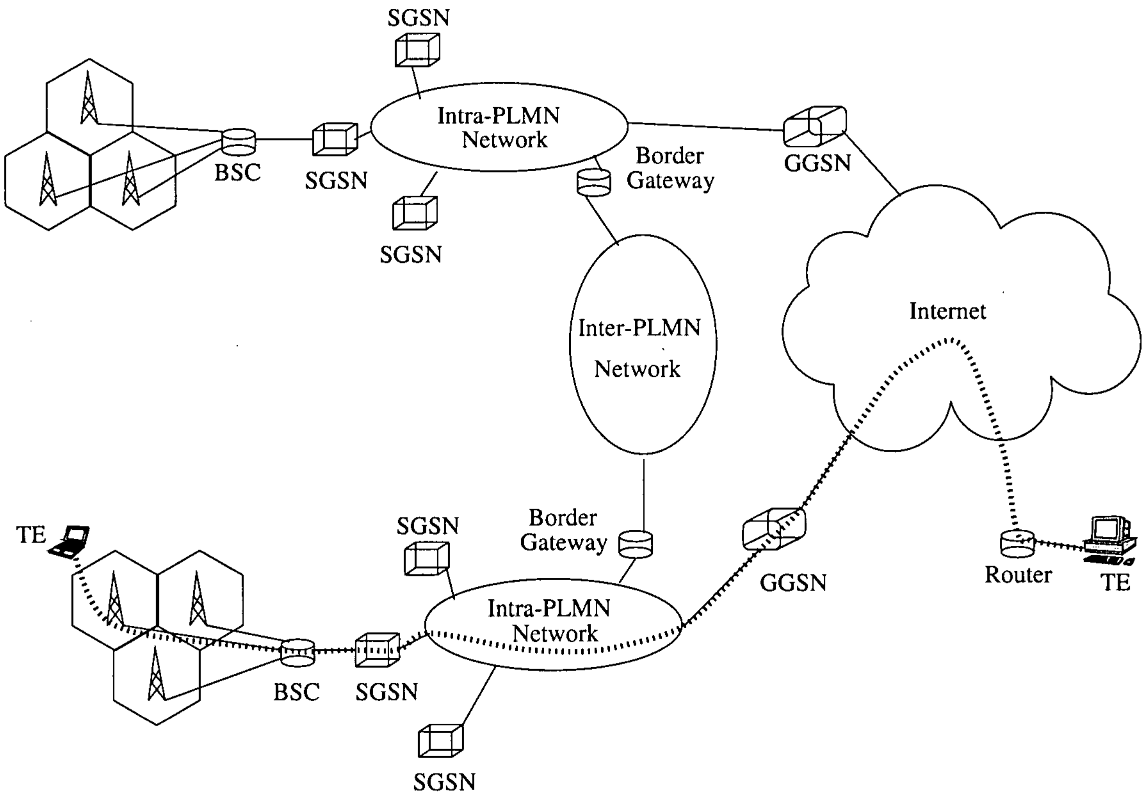

Figure 1.1: The GPRS system architecture

1.2 Research Goals

Providing QoS support to the backbone network by implementing the DiffServ model necessitates resolving a number of issues. First, a proper set of DiffServ Per Hop Behaviors (PHBs) needs to be selected that can bear QoS characteristics equivalent to all of the UMTS traffic aggregates. Second, an interworking system, especially a method for mapping the QoS parameters of one bearer service to another, should be devised. Finally, implementation of DiffServ in the UMTS backbone network requires using DiffServ-aware nodes in the backbone. Therefore, selecting a set of proper DiffServ functional components with an appropriate scheme embedded on each element is vital. The above objective is a massive project and will require extensive time and effort to be achieved.

This research presents an overview of the issues related to QoS support in the UMTS backbone network using the DiffServ model. While it provides some information about QoS in the UMTS and DiffServ systems, associated with the above objective, it concentrates on two main studies: the configuration of a DiffServ-based UMTS router, and the QoS mapping function. Specifically, four functional and traffic conditioning elements of a DiffServ router are examined thoroughly: the scheduler, algorithmic dropper, classifier and meter, as follows.

- The Scheduling element: There are a large number of scheduling algorithms presented in the literature. Among those, Frame-based Fair Queuing (FFQ) is an efficient, fair queuing system which has $O(1)$ timestamp computation complexity and seems suitable for implementation in the scheduler element. However, it lacks the ability to assign decoupled delay and bandwidth guarantees, which is an important factor in defining some PHBs needed in a UMTS environment. This thesis presents a novel service discipline, called Decoupled Delay-Bandwidth FFQ (DDB-FFQ). This scheme is based on the FFQ scheme, and therefore, inherits its characteristics. In addition, it allows assigning the delay bound and bandwidth allocation of a PHB independently.

- The algorithmic dropping element: This thesis also studies the implementation of Random Early Detection (RED) on shared-memory buffers. It proposes and compares three RED on shared buffer schemes. Implementing a proper RED on a shared buffer algorithm on the PHBs corresponding to UMTS interactive or background traffic classes will add congestion avoidance capability, while showing a better loss rate and memory utilization performance. The proposed method is especially applicable to the buffer used for PHBs associated with the background traffic class, such as best effort, lower than best effort, and bulk handling PHBs. This is the first time RED on a shared buffer has been studied.

- The mapping function: A study of current UMTS QoS structure and DiffServ standard PHBs is also undertaken and a basic scheme for mapping UMTS traffic classes to DiffServ PHBs is presented in this thesis.

- Meter: The DDB-FFQ service discipline proposed in this thesis utilizes a rate estimator module for its normal functionality. This module is actually a metering element, and therefore, can also serve as a DiffServ meter. Thus, the DiffServ meter component is also covered here.

1.3 Thesis Outline

This thesis consists of the following chapters. Chapter 2 presents some background information related to UMTS and DiffServ systems. It first covers the UMTS infrastructure and its QoS structure, then studies the DiffServ conceptual model, including the components and structure of a DiffServ router.

Chapters 3 to 5 consider four DiffServ functional blocks in more detail. Chapter 3 addresses the algorithmic dropping element by studying the performance of RED algorithms on shared-memory buffers. After a brief study of RED and memory allocation policies, three RED on shared buffer algorithms are proposed and studied. At the end of Chapter 3, a brief analysis of these schemes is presented.

Chapter 4 studies the scheduling block. First, it reviews the requirements of a favorable scheduler. Then, an efficient scheduling algorithm called DDB-FFQ is proposed. Finally, the rate-estimator module used in DDB-FFQ is studied. This module has a twofold responsibility, performing as a metering element as well.

Chapter 5 studies the DiffServ classifier element and QoS mapping function. It gives an overview of the UMTS QoS structure and DiffServ standard PHBs, and addresses a basic mapping system between these two systems.

Chapter 6 focuses on presenting the OPNET simulation models and discussing their results. First, a performance comparison of the proposed RED on shared-memory buffer algorithms is presented. Then, the simulation model and results corresponding to the DDB-FFQ scheduling scheme are provided. Finally, simulation results indicating the performance of a simple DiffServ-based UMTS backbone model are discussed.

Chapters 7 presents the thesis conclusion and possible future work.

Chapter 2

Background Information

Since the premise of this thesis is the QoS support in the backbone network of a GPRS or UMTS system, a brief understanding of the structure and standard specifications related to QoS in UMTS and DiffServ is essential. This chapter presents an overview of, and background information on, the technologies used as the basis of this project, in two main sections: UMTS structure and the DiffServ conceptual model. The first section provides a short description of the GPRS System structure and UMTS/GPRS QoS architecture. The second section presents an overview of the DiffServ system and router configuration.

2.1 An Overview of UMTS/GPRS

GPRS [1][3][4][5] is an enhancement of GSM that reuses its infrastructure to provide a true packet-switched service. This enhancement provides more efficient radio usage, faster setup time, and therefore, higher data bandwidth. These benefits can be achieved only by introducing two new nodes to the system and by upgrading the software of some other GSM components. The most prominent advantage of GPRS is that it provides a smooth transition from a 2G cellular system to a 3G system realization. UMTS is a 3G mobile system designed to be implemented on the GPRS structure. It will introduce a completely different radio interface and a much higher data rate. However, their QoS architecture and network skeleton are identical so, fortunately, all discussion presented here applies to both [2][6]. In this section, first a brief overview of the system structure and background is given and then the QoS structure as defined in the standard is described.

2.1.1 GPRS System Structure

GPRS uses the GSM infrastructure but introduces two new logical nodes, called GPRS Support Nodes (GSN), for packet transfer within the Public Land Mobile Network (PLMN) [7]. Figure 1.1 illustrates a GPRS system which includes GSM components: Mobile Station (MS), Base Transceiver Station (BTS), Base Station Controller (BSC) ; and GPRS added nodes: Gateway GSN (GGSN), and Serving GSN (SGSN). Figure 2.1 depicts only the structure of the backbone network [8]. BTS handles the radio interface connecting to the mobile station (MS). BSC controls a number of BTSs and manages radio resources and handovers.

SGSN in GPRS is the equivalent of the Mobile Switching Center (MSC) in GSM and is responsible for packet transmission to and from the MS within its service area. GGSN is the system gateway to external packet data networks and is connected to SGSN through an IP-based backbone network. Figure 2.1 shows that the backbone network itself consists of a number of inter-PLMN and intra-PLMN networks. An intra-PLMN network connects the SGSNs and GGSNs of a local networks and an inter-PLMN network connects the intra-PLMN networks together through border gateways.

2.1.2 UMTS/GPRS QoS Architecture

The QoS structure of UMTS has been specified in an ETSI specification[2] that presents a framework for QoS within UMTS. Since the QoS of GPRS and its network structure has been defined identically to UMTS, the following discussion is valid for both of these 2.5G

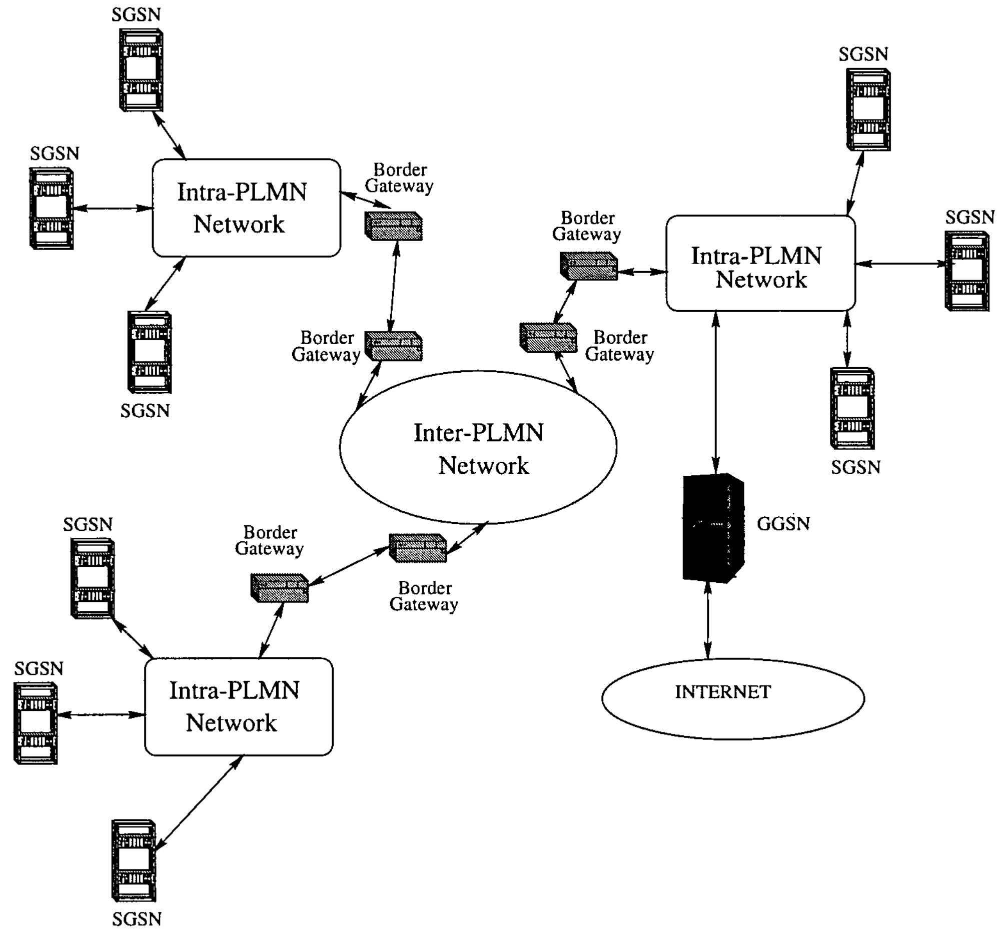

Figure 2.1: A schematic of GPRS backbone network structure

and 3G systems.

What really matters in the QoS support, is what end users observe. This is the QoS as seen from one Terminal Equipment (TE) to another. It is requested by users from a number of finite QoS levels, and it is the user who determines if she is satisfied with the service.

The data path from one TE to another can be divided into a number of bearer services. These bearer services and levels are usually different in nature and may be governed by distinct providers. Therefore, the study of QoS is often easier if examined within each separately.

Figure 2.2 shows the UMTS bearer services layered structure. Each bearer service includes all the features needed for provisioning a contracted QoS. The traffic going from

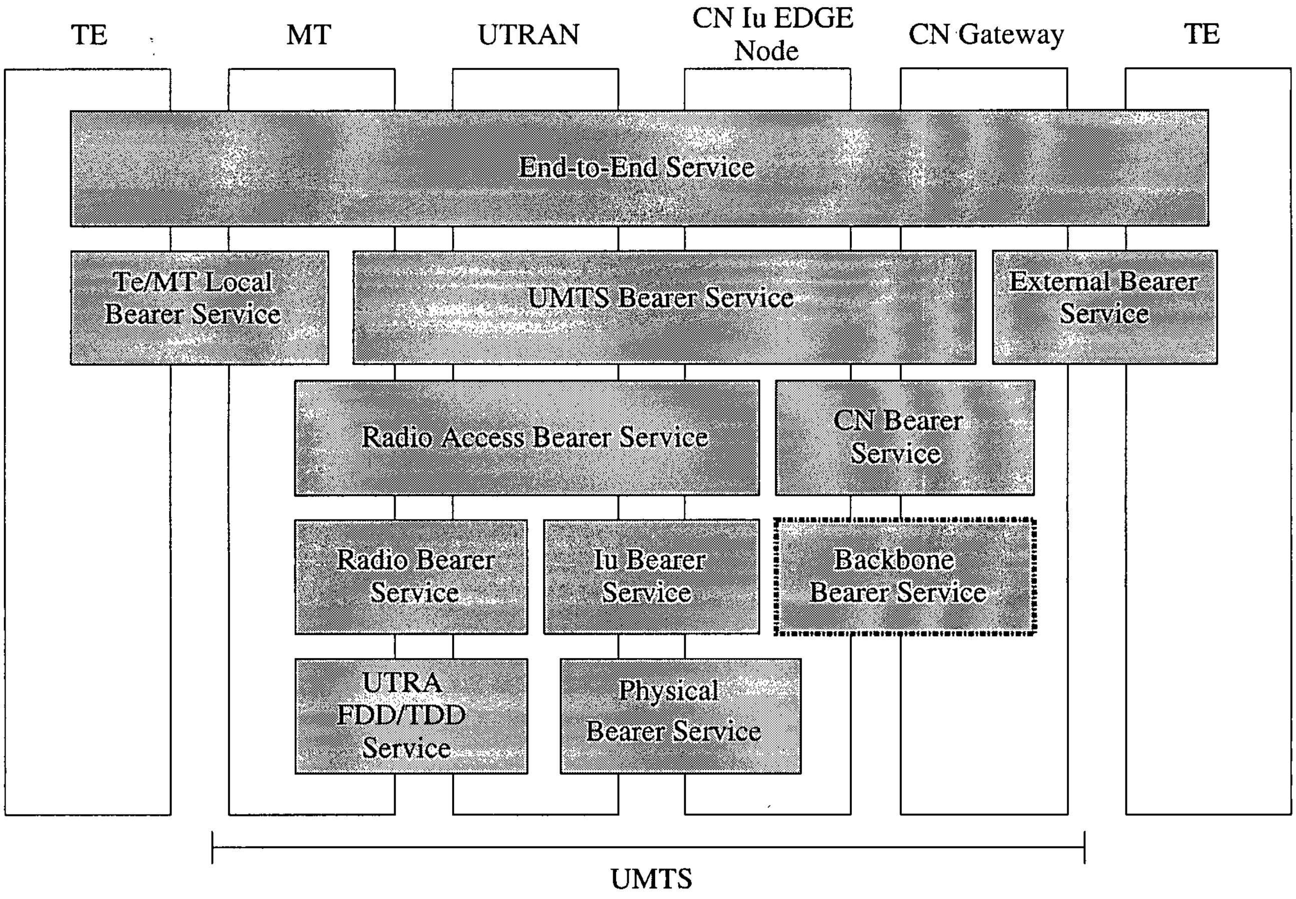

Figure 2.2: UMTS QoS architecture

one TE to another TE passes a number of these bearer services, which can be categorized into three distinct types: TE/MT local bearer service, UMTS bearer service, and external bearer service. A TE is connected to UMTS through a Mobile Termination (MT) and it is set by the user, hence, the TE/MT local bearer service is outside the scope of the UMTS system. The external bearer service is any network or system in the data path that is not included in the UMTS system. Therefore, it is also not elaborated on by the UMTS specification.

The UMTS bearer service which provides the UMTS QoS is itself layered into two parts: radio access and core network. The radio access bearer service is responsible for provisioning an optimum and confidential transport of user data from MT to the core network Iu edge node, with QoS negotiated with the user, which is realized by a radio bearer service and Iu bearer service. The role of the radio bearer service is the radio interface transport and it uses UTRA FDD/TDD [2]. The core network is the main subject of this research and is based on a generic IP backbone network.

UMTS QoS Classes

QoS is conveyed in the form of a single attribute with multiple parameters. It is negotiated with the user as a part of the Packet Data Protocol (PDP) context at the time of connection establishment phase. Due to air interface restrictions, the QoS classes are not meant to be complex, while covering the range of all potential applications. The most noteworthy of UMTS QoS parameters is the traffic class. This is the attribute that defines the type of the stream and other parameters describe the traffic more specifically. Chapter 5 elaborates on the UMTS QoS in more detail.

2.2 DiffServ Architecture

Differentiated Services (DiffServ, DS) is a scalable service differentiation proposed by IETF for IP networks [9]. It achieves scalability by performing traffic flow aggregation. Each aggregate is identified by a code and each node in the way of packet data path treats that packet according to a role defining the Per Hop Behavior (PHB) corresponding to that code. Therefore, no per-application flow or per-user forwarding state is maintained in this structure. In addition, sophisticated traffic management and conditioning tasks are performed only at the edge nodes of a DiffServ network. A DS domain is a DiffServ network consisting of a number of adjacent nodes with the same set of service policies and PHB definitions. A node in a DS domain is classified as either an edge or a core node.

The DiffServ architecture works according to a simple model. When a packet enters a DS domain, it is classified into a finite set of behavior aggregates. These behavior aggre-

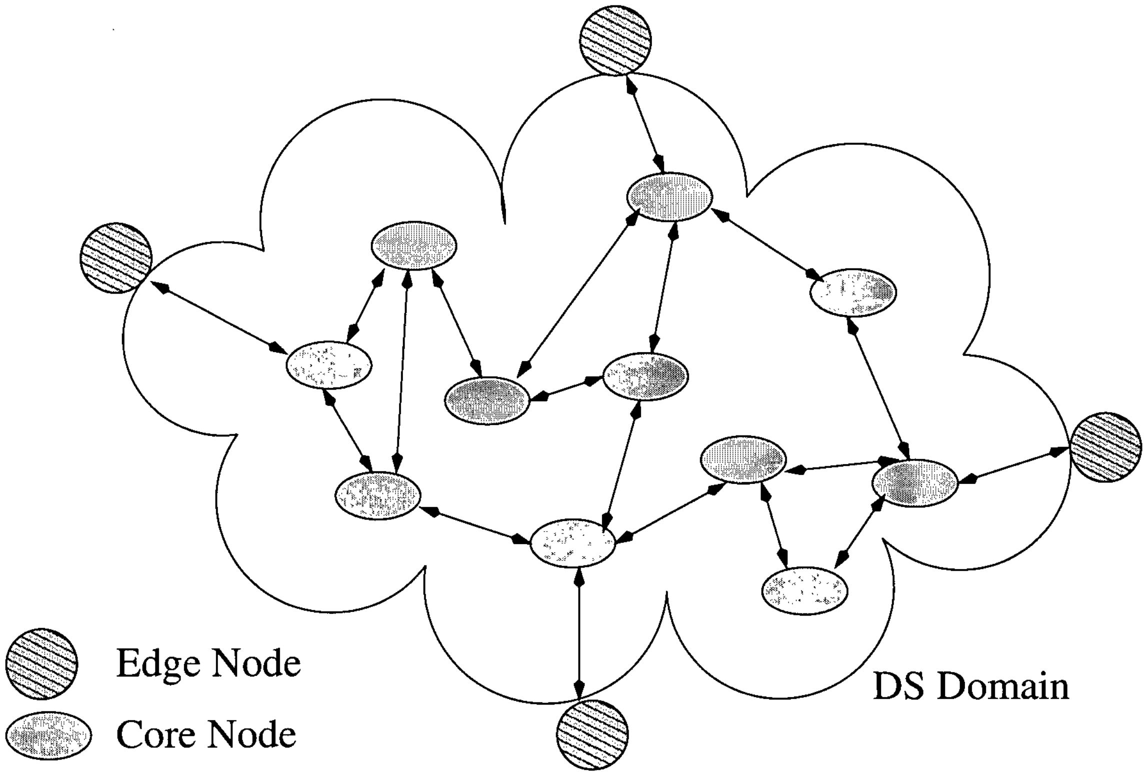

Figure 2.3: DiffServ domain structure

gates are identified by a DS codepoint [10], which is conveyed by marking the header of the IP packet [11]. Any node in that DS domain along the path of that packet treats the packet according to a PHB rule corresponding to its codepoint.

DiffServ is implemented by indicating a small set of per hop behaviors, a classification function, and functional and queuing blocks.

The IETF DiffServ working group has proposed a number of standard PHBs. They are expedited forwarding (EF), assured forwarding (AF), best effort (BE), lower than best effort (LBE), alternative best effort (ABE), dynamic real-time/non-real-time (RT/NRT), and assured rate (AR). Chapter 5 presents a thorough discussion of these schemes.

There are a number of other packet differentiation schemas used in the literature. The most common of these schemes are relative priority marking, service marking, label switching, and integrated services/Reservation Protocol (RSVP). Hereafter, DiffServ will be concisely contrasted with each of the above methods [9].

In a relative priority marking model, packets are assigned a precedence for one of their characteristics, such as: delay, bandwidth, and packet loss and are prioritized accordingly. An example of this model is 802.3 Token Ring priority. DiffServ can be considered a generalization and refinement of this scheme, since it utilizes general PHB concepts and propose a whole architecture.

An example of service marking is IPv4 Type of Service (ToS) [12], used for routing purposes. In this scheme packets are marked for minimizing delay or maximizing bandwidth to direct the routers along its path in selecting the route. This scheme is totally different to DiffServ, since DiffServ is not concerned with routing.

Examples of label switching are ATM and MPLS. In this model, a path is established in each node along the packet path from the source to the destination. Therefore, each node needs to maintain a QoS state associated with this path. Although this model provides a finer QoS support, it has extra management and configuration requirements, and also a scalability problem, that is, the number of states in a node grows with the number of established paths.

In an RSVP model, the source, the destination and the nodes along the path all exchange signaling messages to establish a packet classification and forwarding state in each of these nodes. This model also has a scalability problem, if no state aggregation is performed.

2.2.1 DiffServ Router Model

This section describes the elements used in a DiffServ-aware node. Figure 2.4 illustrates an elaborate component interaction of the high-level view of a DS node [13]. The routing core is an abstract of a router normal routing functionality and is not discussed in DiffServ model. The configuration and management interface monitors service provisioning and gathers statistics regarding the traffic of each service level. These are used as network management bases from which DiffServ policies are assigned or altered. The network administrator interacts with this module through a management protocol. The optional QoS agent is considered to add per-flow and per-flow-aggregate signaling capability to DiffServ model. This allows to snoop RSVP messages.

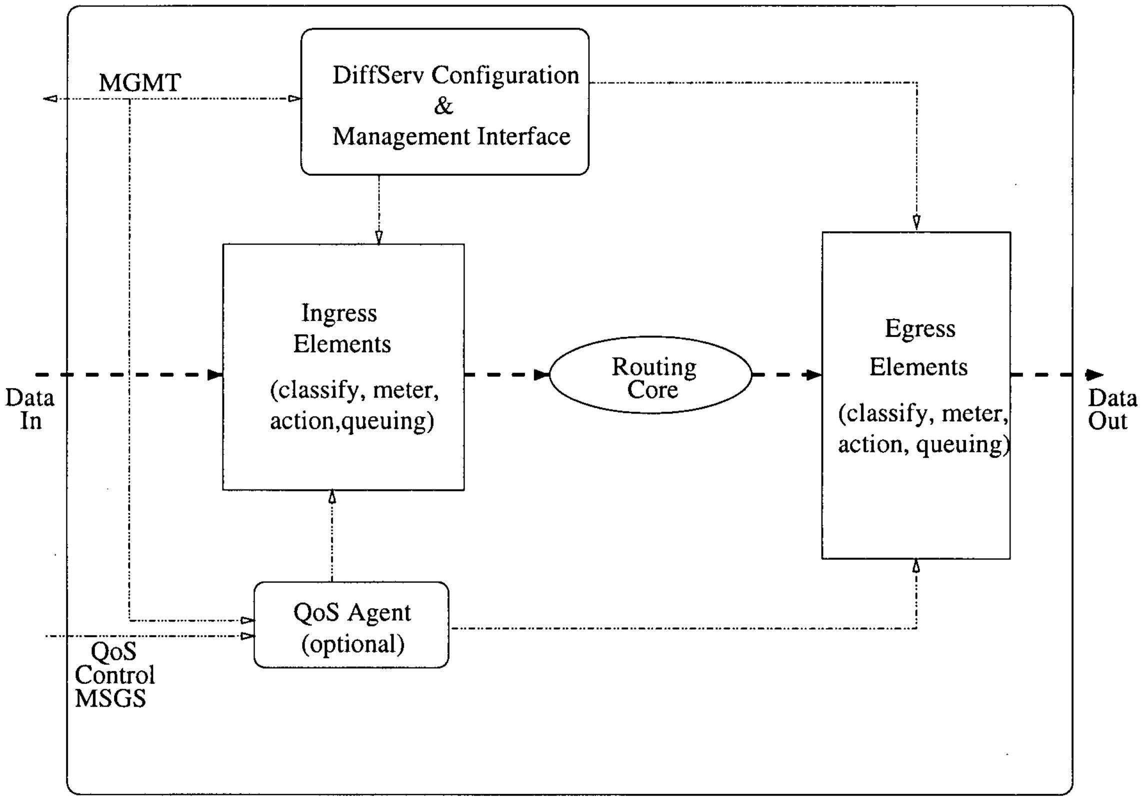
MGMT
DiffServ Configuration & Management Interface
Ingress Elements (classify, meter, action,queuing)
Routing Core
Egress Elements (classify, meter, action, queuing)
Data In
Data Out
QoS Agent (optional)
QoS Control MSGS

Figure 2.4: A DiffServ router's major functional blocks

Figure 2.5 shows a node with two egress/ingress interfaces. It is possible to have an arbitrary number of these interfaces in an actual router. Also it is not mandatory to implement all of these blocks and their components on both ingress and egress points. The configuration of components depends on the service requirement of a router.

Figure 2.6 shows the datapath that a data packet might take in a router. On arrival, a packet is first classified according to a set of rules. Then it is checked for being within its allocated rate by a metering block. Afterward, the packet is passed along a set of traffic

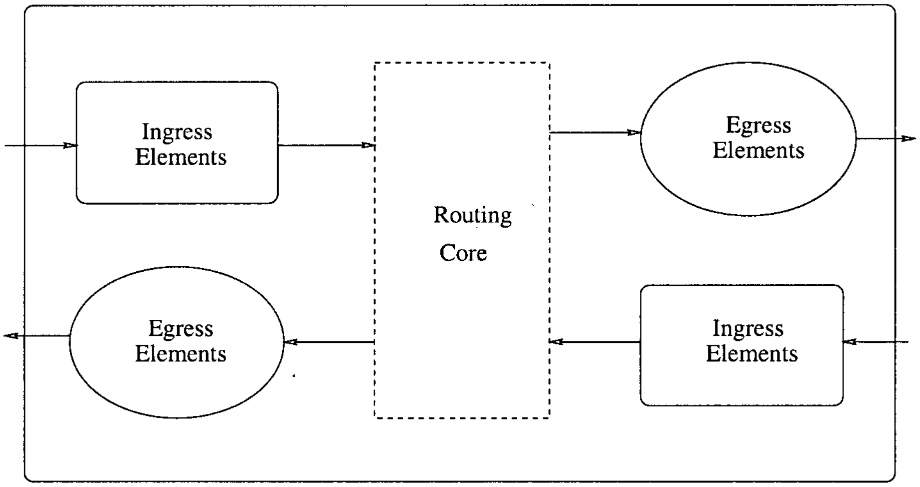

Figure 2.5: The structure of a DiffServ router

conditioning blocks such as a marker, dropper, counter and if accepted, is enqueued in the queuing block and then transmitted within a scheduler policy.

There are four kinds of components in an ingress/egress interface: a traffic classifier, meters, actions and queuing elements [14]. Action elements are markers, droppers, counters, and multiplexers.

Mapping Function: Mapping is the function of translating the QoS of one system to the QoS parameters understandable by another QoS system. It is a necessary function for a UMTS packet entering the DiffServ-aware network, or when a packet coming from an external network enters the UMTS network through a GGSN. Chapter 5 gives a thorough discussion of this topic. This function is usually integrated with the classifier into one functioning block.

Classifier: A Classifier is a functional element that selects packets, based on policy. It can classify packets based on the content of the packet header, other packet data and/or some implicit information related to that packet. In a DiffServ model, the packets are classified according to the content of a field in the IP header, originally called Type of Service (ToS) in the IPv4 protocol. ToS is an 8 bits field and DiffServ uses 6 bits of it, also called a

DS field, to convey the DiffServ CodePoint (DSCP). The remaining 2 bits, called CU (currently unused), are not used in DiffServ and are proposed for use in congestion notification function by IETF Explicit Congestion Notification (ECN) group. The classifier has a single input and splits the incoming stream into a number of outgoing ports. The most common way of implementing classifiers is to use a number of filters. A filter is a set of match conditions based on a classification key. The contents of the DSCP is passed through the filter, and according to the matching results, it is grouped into a PHB aggregate.

Meter: A meter is a functional block responsible for measuring the rate characteristics of a packet stream. These gathered statistics are used by other blocks for their decision making. For example, a marker uses this data to re-mark the packet DSCP. IETF has a set of RFCs proposing two main meter implementation: the time sliding window meter/marker and the two(three) token bucket meter/marker.

Marker: This module is responsible for setting the DS field of a DiffServ packet according to various rules. It may also use the statistics gathered by a meter to decide on the level of a packet's conformance to its allocated rate characteristics. For a non conformant stream, it takes action by re-marking the packet in a class with lower priority, or in a class with a lower drop precedence. It is also responsible for assigning a valid DSCP to a packet with an unknown DSCP.

Counter: A counter is responsible for updating the packet counter. It is also used to count the number of packets passing through it.

Dropper: There are two major kinds of dropping elements in a DiffServ router: an absolute dropper and an algorithmic dropper. While absolute droppers are quite straightforward, research is still being undertaken on the viability, usability and the schemes of algorithmic droppers. Random Early Detection (RED) is a well-known algorithmic dropper, suggested for avoiding congestion in the network. There are a number of varieties of RED. In this project, the implementation of RED on shared-memory buffers is addressed and proposed for deployment on a DiffServ router for TCP-compatible aggregates.

Queuing Block: This is the block that stores the packets while they are waiting to receive service by the scheduler and depart to the next hop or DiffServ module. There are two different types of buffers used for enqueuing packets: shared memory buffers and dedicated buffers. In a dedicated buffering method, each queue has its own separated memory space, while a shared buffering scheme uses a large memory which is shared among different traffic classes, according to a sharing rule. Chapter 3 presents a thorough study of shared-memory buffers.

Scheduler: The scheduling block has the most impact on the level of service a packet receives. It decides on which queue, among a set of queues, to service next. There are a large number of scheduling schemes addressed in the literature, each of which has some advantages and some disadvantages. In a DiffServ-based UMTS backbone network, the most desirable scenario is to have a fair, efficient and simple scheme which supports link sharing and delay bounds. It should also be able to assign different delay bound guarantees independently of bandwidth allocation, and vice versa.

Figure 2.6 illustrates an example of a DiffServ router using all of the above components. This router supports four PHB aggregates. The first group is an EF PHB. This traffic is metered first and non-conforming traffic is dropped, but the conforming traffic has a guaranteed delay bound. The second group is an AF11 group. It has a dedicated buffer that accepts traffic packets based on a tail dropping scheme. This traffic can be video streaming traffic. The third group is an AF21 group that first meters the traffic flow. If the traffic is non conforming it is re-marked for an AF22 (with a lower drop precedence). The last group is best effort traffic. Groups three and four use a shared-memory buffer for their queue and a RED on shared buffering scheme is applied as the dropping scheme.

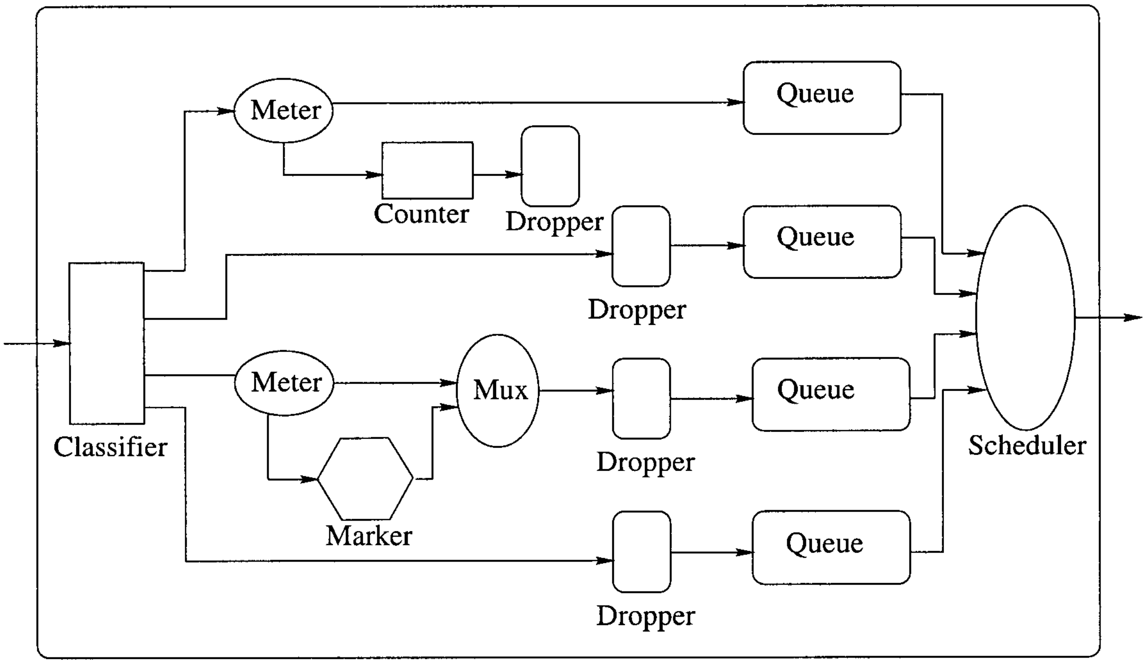

Figure 2.6: An example of traffic conditioning blocks in a DiffServ router

Chapter 3

Random Early Detection (RED) Algorithms for Shared Buffers

3.1 Introduction

Two kinds of dropping elements are used in the DiffServ model: the absolute dropper and the algorithm dropper. An absolute dropper simply destroys its received packet. It is used to drop the non conforming traffic of those applications that do not have an end-to-end congestion manager and are overloading the network. An absolute dropper is also used to model tail dropping queues that drop the packet only when full. However, algorithmic dropping is an efficient way of controlling and avoiding congestion. Congestion happens when the rate of the incoming traffic surpasses the service rate capability of a router or a link. The Internet experienced its first set of crashes, known as congestion collapse, in October 1986 [15]. This event led to the implementation of a congestion control mechanism in TCP, the main transport protocol in use at that time, to maintain Internet functionality.

Even in a normal data network, the traffic is "bursty" and changes continuously. To absorb this bursty traffic, buffers are used in each intermediate node along the traffic path.

A bigger buffer can capture a larger amount of data, but it will impose a longer average delay. However, no matter how large a buffer is, at the time of congestion it becomes full and some packets should be dropped. Besides, Internet traffic is increasing continuously. If a router behaves proactively and informs sources that it is at the threshold of becoming congested, congestion can be avoided. This idea led to the introduction of congestion avoidance algorithms.

There are two kinds of congestion control methods [16][17]: end-to-end congestion control and network (router) based congestion control. In the former method, only the two end-sides of the connection are responsible for using the network up to capacity, without overloading it. An example of this type is the method deployed in TCP, which has been preventing congestion collapse in the Internet for a long time. TCP is the only transport protocol equipped with a congestion control mechanism. However, there are many recent applications that use UDP, and if their traffic occupies a substantial percentage of Internet traffic, history might repeat itself. Luckily, IETF has a working group on a universal congestion manager and has proposed a general congestion manager interface to add this capability to any kind of application [18]. In a system with only end-to-end congestion control, intermediate nodes will accept every packet they received, up to their memory capacity and drop packets when their buffers become full. This kind of packet dropping is called "tail dropping" throughout this thesis. This term is also used in the literature as a contrast to "head dropping", indicating whether the packet at the head of a queue is dropped, or one at its tail.

The Internet has been relying on end-to-end congestion control methods for a long time, but this is inadequate for a number of reasons. One, TCP functions in a way that keeps queue occupancy high. This imposes discrimination in drop probability for bursty traffic, since an arriving burst of data will not find enough space in the buffer. Two, TCP itself is assumed to be bursty. Three, there are many existent or emerging applications that do not obey any kind of end-to-end congestion control mechanism [19].

With network based congestion control, every node in the path also acts proactively to keep the usage of the network within its capacity. The most prominent of these schemes is Random Early Detection (RED). RED was first introduced in [16] for traditional single queue gateways to add congestion avoidance capability to the routers. It detects the incipient congestion and drops/marks packets before the queue becomes full. By dropping/marking packets, it notifies the source to decrease its traffic rate to prevent congestion [16][19]. Marking packets is a more logical way of informing the corresponding source, and is an ongoing focus of research. However, its implementation needs a corresponding transport protocol that understands this method. TCP has been using packet drops as a way of evaluating network conditions. Therefore, currently, packet dropping is the main consideration, while packet marking is under study and proposed in an RFC [20]. These packet drops match properly with TCP-compatible transport protocols and prevent severe source throughput degradation. RED is standardized by the Internet Engineering Task Force (IETF) in an RFC [21] as a better alternative of the traditional tail drop mechanism.

Deployment of RED has a number of advantages. One, it tries to keep the buffer occupancy low, therefore causing a lower average delay. Two, it prevents concurrent dropping of large numbers of packets in bursty traffic. Three, it does not have a bias against bursty traffic. Four, TCP reacts more severely to burst packet losses than to single packet loss.

3.2 RED on a Single Queue

In a gateway with a RED [16] buffer management scheme, packets are dropped/marked before its buffer gets full. The objective of RED is to detect incipient congestion and take measures to avoid it. It is intended to be used for networks in which the transport protocol reacts to congestion notification feedback from the network. Thus, the RED gateway has to perform two tasks. It should first apply an algorithm for detecting the congestion, just before it happens, and then it should take action by applying another algorithm for its congestion avoidance task.

RED uses the average queue size of the router as an indicator of the congestion intensity level. Although there are a number of ways of calculating the average buffer size, Exponential Weighed Moving Average (EWMA) is the most common method. EWMA is calculated by adding a weighted value of the current queue size to the weighted value of the previous average queue size, putting more weight on the current data. In an EWMA filter, the average queue occupancy, *avg*, is estimated by

$$avg \leftarrow avg * (1 - \alpha) + k * \alpha. \tag{3.1}$$

In this equation, *k* is the current size of the buffer and α is the weight factor. α is the key parameter for setting the time constant of this lowpass filter and its value is always less than one. When it is set to a value of greater than 0.5, it puts more weight on current data and when it is set to a value of less than 0.5, it puts more weight on past data.

RED calculates the average queue size for every incoming packets. Having computed the average queue size, its uses the following algorithm to decide on dropping/marking that packet. It uses two threshold values called the maximum threshold, *max-th*, and the minimum threshold, *min-th*. If *avg* is less than *min-th*, the packet is accepted and enqueued. If *avg* is greater than *max-th*, it will be dropped/marked. However if the value of *avg* is in between *min-th* and *max-th*, the dropping probability p_k is first calculated thusly:

$$p_k = max_{prob} * \frac{avg - min_{th}}{max_{th} - min_{th}}. \tag{3.2}$$

Then the packet is dropped with the probability of p_k. max_{prob} is a parameter that determines the maximum value of the dropping probability.

In other words, in a RED gateway, packets are dropped with a probability which is based on a function of the average occupancy size of that buffer, possibly well before it gets full. Figure 3.1 shows the dropping probability versus average queue size of a RED gateway with the physical buffer size of B, maximum and minimum thresholds of *max-th* and *min-th*, and maximum drop probability of max_{prob}.

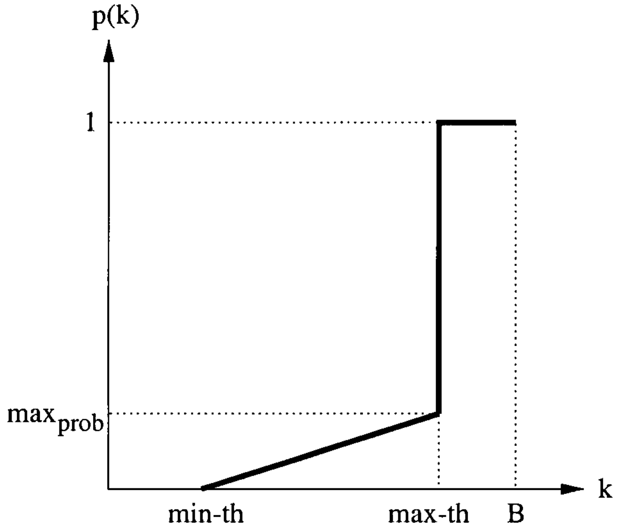

Figure 3.1: RED dropping mechanism

The pseudo code of the RED algorithm is as follows [16]:

```
for each packet arrival
    calculate the average queue size (avg)
    if min-th $\leq$ avg $<$ max-th
        calculate drop probability $p_k$
        with probability $p_k$ mark the arriving packet
    else if max-th $\leq$ avg
        mark/drop the arriving packet
```

3.3 Multi-Class RED

3.3.1 Introduction

The current Internet treats every packet in a best effort manner. Therefore, each node has only one queue for all incoming packets. In a class-based buffer, distinct queues are used to keep packets of different classes separate. There are two buffer allocation policies for implementing a class-based buffer: the shared-memory and the dedicated-memory policies. In a dedicated memory buffer, a completely separate queue is used to store the packets of one traffic class. This scheme has the advantage of providing guaranteed space for each class and requires less queuing administration. However, in a shared-memory buffer, all the packets share the same buffer. A sharing policy allocates enough memory space for an incoming packet and keeps track of packets belonging to each class. This scheme provides more efficient memory utilization, since the unused memory space of one class can be used by another. This allows the buffer to have higher bursty traffic handling capability.

In a DiffServ router, congestion control is currently only applicable to non-real time traffic, until such time as congestion manager modules are used for all kinds of traffic[18]. Therefore, mainly interactive and background traffic classes need proactive congestion control. Each of these traffic classes are mapped to a number of PHB groups. Therefore, a shared buffer can be used to store packets of these groups. This section proposes and studies the congestion control schemes applicable for a shared memory buffer. Using a shared-memory buffer has a number of advantages for these PHB groups:

- The memory is used more efficiently. In a dedicated-memory system, the router may become congested for one or a few traffic classes while the queues of other classes are not completely used. However, in a shared buffer, the unused memory of a class may be used by the other classes to handle short-term congestion.

- A shared buffer is managed by software and provides soft division for each queue. For a DiffServ router in which the number of its supported PHB or its parameters are to be modified or to be set on demand, using sharing schemes are very convenient. An example of software based routers is given in [22].

- Although the algorithms presume symmetric traffic flow, by applying a weighted version of these algorithms and using push-out, the system also works in a fair manner for non symmetric traffic. It is also easy to configure the system to work in favor of a particular traffic class. Push-out is the act of freeing enough space to store an incoming packet when the queue is full, by dropping some packets that have already been accepted from a lower priority, or from a queue that has used the memory unfairly.

- Finally, the simulation results presented in Chapter 6 show better performance in the deployment of RED on shared buffers, in terms of packet loss.

However, shared buffer schemes are more complex compared to dedicated systems. They require a more elaborate and sophisticated buffer manager. They also need to deal with packet allocation and resumption. For non-static threshold systems, performing push-out and/or applying dynamic policies adds to the complexity of the system.

The following sections are focused on the congestion control schemes applicable to shared-memory buffers. First, a short overview of current buffer sharing schemes is presented. Then, three different deployments of RED on shared-memory buffers that have been derived from present buffer sharing schemes, are proposed. These schemes work both as sharing policies and as algorithmic droppers for the purpose of congestion avoidance. Finally, an analytical representation is provided.

3.3.2 Memory Sharing Policies

A shared-memory buffer is a single pool of memory shared by the packets of a number of traffic streams, while they wait to be serviced and transmitted. A buffer allocation policy is applied to enable proper queuing functions. In a sharing system, the sum of the individual partitions is equal to the the total memory. Hereafter, sharing schemes addressed in the literature are summarized [23] [24] [25].

Complete Partitioning (CP): In a complete partitioning sharing scheme, the entire buffer is permanently partitioned among the classes. Therefore, CP actually does not provide any sharing. However, since the queues are not physically isolated, it is also classified among sharing schemes. This scheme can be considered equivalent to a dedicated memory system, in which separated queues are used for different classes and has most of its properties, such as packet loss rate. Thus, another advantage of studying this scheme is that it enables us to make a performance comparison between dedicated and sharing systems. Still, CP has an important advantage of shared buffer systems in that it is easily reconfigurable, but it also has the disadvantage of complex buffer management.

Complete Sharing (CS): Complete sharing lies on the other extreme, compared to CP. In this scheme, the buffer is shared among all queues, without limitation. Every incoming packet is accepted, as long as the buffer is not full. In a CP policy, the buffer allocated to a queue may be wasted if its corresponding traffic class is inactive, while in the CS policy, a buffer may be monopolized by a traffic class. However, CS policy efficiently uses the memory and can be fair, if the traffic is symmetrically divided into all classes.

Sharing with Minimum Allocation (SMA) : This policy is a combination of CS and CP policies. In the SMA scheme, a minimum memory space is put aside for each class and the rest is shared between all the arriving packets. Therefore, each traffic class has a minimum guaranteed memory space and the remainder memory is efficiently used by all classes.

Sharing with MaXimum Queue lengths (SMXQ) : This scheme is similar to the CS policy, but with the added feature of a maximum limitation on the memory space occupied by each traffic class. SMXQ prevents a traffic class from monopolizing the whole buffer. The values of these maxima are considered so that their sum is greater than the whole memory size.

Sharing with MaXimum Queue and Minimum Allocation (SMQMA): SMQMA is the most complex buffer sharing policy and utilizes a combination of SMA and SMXQ features. In this scheme, the buffer is shared among all traffic classes, but with a reserved minimum space, and a maximum space limit for each. Therefore, while providing a guaranteed minimum space, it also restricts a class from monopolizing the whole shared memory space. On the down side, it requires the largest number of state variables for its performance and has the highest complexity.

3.4 RED on a Shared Memory Multi-Class Buffer

In this section, the deployment of RED on shared memory buffers is addressed. Based on three of the sharing policies discussed previously, three corresponding RED algorithms are proposed. These algorithms perform two tasks. First, they provide buffer allocation policy similar to the sharing scheme they are derived from. Second, they provide a RED-like congestion avoidance functionality. In these schemes, the RED maximum and minimum thresholds, which also determine the maximum and minimum sharing limits, are defined both for the whole buffer and for each queue.

Based on complete sharing, complete partitioning, and sharing with minimum allocation, three algorithms are proposed. They are RED on complete sharing (R-CS), RED on complete partitioning (R-CP) and RED on sharing with minimum allocation (R-SMA).

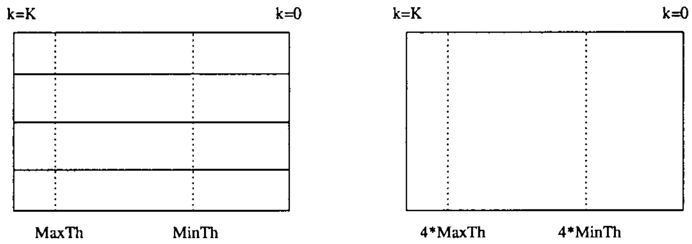

Figure 3.2: Comparison of RED thresholds in RED-CP and RED-CS schemes

During this discussion the following terminology is used. "Buffer" or "whole memory" or "queue" indicates a single large memory shared by all the subqueues of different classes. The memory space used for storing packets of the traffic class *i* is called "subqueue *i*". The RED average queue size calculated for the whole memory is indicated by *avg* and the RED average queue size of subqueue *i* is indicated by *avg(i)*. Similarly the minimum and the maximum thresholds for the whole memory are called *max-th and min-th* and those of subqueue *i* are indicated as *max-th(i)* and *min-th(i)*, respectively.

3.4.1 RED on Complete Partitioning (R-CP)

In RED on complete partitioning, the memory is permanently divided among the subqueues, that is, each traffic class has its own fixed memory space and a traditional RED algorithm is applied on each subqueue independently of the other queues. This scheme is identical to a multi-class dedicated-memory queuing system, from a performance viewpoint. Therefore, it has also been used as the basis for comparing the performance of RED on shared and dedicated memory buffers. This scheme is suitable for those situations when it is preferable to have isolated subqueues, while having some of the advantages of shared memory systems, such as reconfigurability.

3.4.2 RED on Complete Sharing (R-CS)

Based on the complete sharing policy, RED on the complete sharing is defined with the following pseudo code:

```
for each packet arrival with class i
    calculate the average queue size avg
    calculate average sub-queue i size avg(i)
    if min-th $\leq$ avg $<$ max-th
        if min-th(i) $\leq$ avg(i)
            calculate drop probability $p_a$
            with probability $p_a$ mark the arriving packet
    else if max-th $\leq$ avg
            mark/drop the packet
```

In this algorithm, the whole buffer is shared among all the subqueues. For an incoming packet with class *i* the average queue and subqueue *i* is calculated. If the average queue size is less that the minimum threshold, the packet is accepted. Otherwise, the average subqueue *i* size is compared to *max-th(i)* and *min-th(i)* and a decision to accept or discard the packet is made, similar to a traditional RED algorithm.

3.4.3 RED on Sharing with Minimum Allocation (R-SMA)

Here, another RED for shared memory buffers based on sharing with minimum allocation is proposed, called RED on sharing with minimum allocation. This algorithm defines a minimum memory allocation space for subqueue *i* by using RED *min-th(i)*. The pseudo code of this scheme is as follows:

```
for each packet arrival with class i
    calculate the average queue size avg
    calculate average sub-queue i size avg(i)
    if min-th(i) $\leq$ avg(i) and avg $<$ max-th
        calculate drop probability $p_a$
        with probability $p_a$ mark the arriving packet
    else if max-th $\leq$ avg
            mark/drop the packet
```

The deployment of RED on the SMXQ sharing scheme has not been defined here, since it is apparent that there is no other way of implementing RED on it, other than independently applying RED on each subqueue.

3.5 Analysis of RED Mechanism

In this section, an analytic study of RED on shared-buffer algorithms is presented. First, the packet drop probability of a tail drop queue is examined and then the analytical representation of RED on a single queue, RED on a multi-class buffer with complete sharing (R-CS), and RED on a multi-class buffer with complete partitioning (R-CP) are addressed. In the analysis, the following assumptions have been made for the queuing system and the traffic model:

- Packet arrival time is in Poisson distribution, that is, packets have exponentially distributed interarrival times.
- Packet departure rate has Poisson distribution, that is, packet lengths are exponentially distributed.

- Physical buffer size is *K*.

- Stationary state probability of being in state *k* is $\pi(k)$. Being in state *k* means there are *k* packets in the system.

The exponential distribution has been used to allow us to use the PASTA (Poisson Arrivals See Time Averages) property [26]. PASTA property claims that for a Poisson arrival, *the state probability distribution seen by a new arrival would be the same as the time averaged state probability distribution.* In other words, the state probability distribution is equal to the fraction of time that the system is in that state.

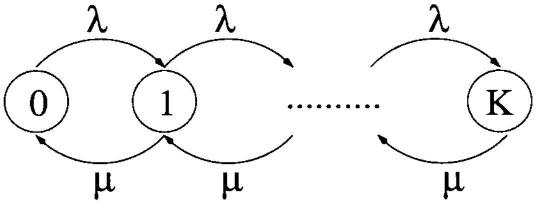

Figure 3.3: M/M/1/K queuing system

3.5.1 Tail Dropping Queue

Figure 3.3 shows an M/M/1/K queuing system. If λ is the packet arrival rate and μ is the packet departure rate, the utilization is defined as: $\rho = \frac{\lambda}{\mu}$. For a buffer with tail dropping and buffer size of *K*, the probability of being in the state *k* is [27]:

$$\pi(k) = \begin{cases} \pi(0) * \rho^k & \text{if } k < K, \\ 0 & \text{otherwise.} \end{cases}$$

By using the normalizing condition, that is, $\sum_{k=0}^{K} \pi(k) = 1$, $\pi(0)$ the probability of being at state zero is simply calculated. Thus:

$$\pi(0) = \frac{1}{\sum_{k=0}^{K} \rho^k} = \frac{1}{\frac{1-\rho^{K+1}}{1-\rho}} = \frac{1-\rho}{1-\rho^{K+1}}. \tag{3.3}$$

The dropping probability can be also calculated by using the PASTA property. According to PASTA, the probability of dropping a packet in a tail drop system is equal to the probability of being at state K. Therefore:

$$P_{TD} = \pi(K) = (1-\rho)\rho^K. \tag{3.4}$$

This equation enables us to have a consistent representation of drop probability for all the schemes.

3.5.2 RED on a Single Queue

For a queue with RED buffer management scheme, a packet gets dropped depending on the value of the drop probability *d(k)* and the queue state probability $\pi(k)$, where *k* indicates the state. To make analysis possible, the instantaneous queue size has been used instead of the average queue size.

To calculate the packet drop probability, the PASTA property is used. In addition, due to RED algorithm, a packet in state *l, l=1, ..., K* is dropped with probability *d(l)*. Therefore, the packet drop probability is calculated thus:

$$\begin{aligned} P_{R1} &= \pi(K)d(K) + \pi(K-1)d(K-1) + \ldots + \pi(1)d(1) \\ &= \sum_{k=1}^{K} \pi(k)d(k). \end{aligned} \tag{3.5}$$

In this equation *d(k)* is the dropping function of the RED algorithm and can be presented as:

$$d(k) = \begin{cases} 0 & \text{if} \quad \hat{k} < minTh \\ \frac{\hat{k}-minTh}{maxTh-minTh} * max_p & \text{if} \quad minTh < \hat{k} < maxTh \\ 1 & \text{if} \quad \hat{k} > maxTh \end{cases}$$

where $\hat{k}$ is the average queue size calculated by RED algorithm.

The state probability used in the above equations is different from the one used for the tail dropping. To derive the state probability, we use a detailed balance equation that says $\pi(i)p(i,j) = \pi(j)p(j,i)$, where $p(a,b)$ is the state transition from state a to state b [28].

Under equilibrium conditions, flows may be balanced across any boundary, so for the k^{th} boundary, let i= k and j=k+1. For a queue with RED implemented, the probability of a transition from state k to k+1 also depends on the probability of packet acceptance. Therefore, the transition probability from state k to state k+1 is $p(k,k+1) = \lambda_k(1-d(k))$ and by applying it in the balance equation we have this:

$$(1-d(k))\lambda_k\pi(k) = \mu_{k+1}\pi(k+1), \qquad k = 1,2,...,K \tag{3.6}$$

which leads to the state probability

$$\pi(k+1) = \frac{\lambda_k}{\mu_{k+1}}(1-d(k))\pi(k). \tag{3.7}$$

By substituting k instead of k+1 and considering $\{\ \lambda_k = \lambda, \quad \mu_k = \mu, \quad \forall k\ \}$, the state probability $\pi(k)$ is:

$$\begin{aligned}\pi(k) &= \rho(1-d(k-1))\pi(k-1)\\ &= \pi(0)\prod_{i=0}^{k-1}\rho(1-d(i))\\ &= \rho^k\pi(0)\prod_{i=0}^{k-1}(1-d(i)).\end{aligned} \tag{3.8}$$

$\pi(0)$ can be easily found by applying the normalizing equation. Therefore,

$$\sum_{k=0}^{\infty}\pi(k) = \sum_{k=0}^{K}\pi(k) = \sum_{k=0}^{K}\rho^k\pi(0)\prod_{i=0}^{k-1}(1-d(i)) = 1 \tag{3.9}$$

and the probability of being is state 0 is

$$\pi(0) = \frac{1}{\sum_{k=0}^{K}\rho^k\prod_{i=0}^{k-1}(1-d(i))}. \tag{3.10}$$

Finally, by applying (3.10) in (3.8), the state probability of RED is calculated as follows:

$$\pi_{RED}(k) = \frac{\rho^k\prod_{i=0}^{k-1}(1-d(i))}{\sum_{k=0}^{K}\rho^k\prod_{l=0}^{k-1}(1-d(l))}. \tag{3.11}$$

3.5.3 A Multi-class RED on Complete Sharing (R-CS)

All the equations derived for the RED on a single queue, other than the dropping function *d(k)*, are directly applicable to the R-CS scheme. The dropping function is based on the R-CS algorithms and is expressed as follows:

$$d(k) = \begin{cases} 0 & \text{if } \ \hat{k} < minTh \\ \left[\begin{array}{lll} 0 & \text{if} & \hat{k}_i < minTh_i \\ \frac{\hat{k}-minTh}{maxTh-minTh} * max_p & \text{if} & minTh_i < \hat{k}_i \end{array} \right. & \text{if } \ minTh < \hat{k} < maxTh \\ 1 & \text{if } \ \hat{k} > maxTh \end{cases}$$

where $\hat{k}$ is the average buffer size and $\hat{k}_i$ is the average size of queue i calculated by RED algorithm.

With the new dropping function *d(k)*, The state probability and the drop probability of an R-CS scheme are written as follows:

$$\pi_{R-CS}(k) = \frac{\rho^k \prod_{i=0}^{k-1}(1-d(i))}{\sum_{k=0}^{K} \rho^k \prod_{l=0}^{k-1}(1-d(l))}, \tag{3.12}$$

$$\begin{aligned} P_{R-CS} &= \pi(K)d(K) + \pi(K-1)d(K-1) + \ldots + \pi(1)d(1) \\ &= \sum_{k=1}^{K} \pi(k)d(k). \end{aligned} \tag{3.13}$$

3.5.4 A Multi-class RED on Complete Partitioning (R-CP)

This case is similar to having a set of independent single queues where traffic is divided between them. Therefore, the equation derived for a single RED queue is applied to each of the R-CP subqueues. Figure 3.4 shows the model of the R-CP scheme for a traffic consisting of four Poisson flows, λ_0 to λ_3. By utilizing the Poisson stream property, the combination of traffic also has a Poisson distribution with an average arrival time of $\lambda = \lambda 0 + \lambda 1 + \lambda 2 + \lambda 3$. This fact is also true for the service rates μ.

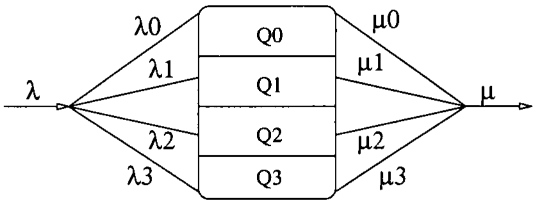

Figure 3.4: Poisson traffic property that can be split or combined

3.6 Comparison Between Tail Drop, R-CS, and R-CP

For R-CP scenario, we consider a symmetric system with $\lambda_i = \frac{\lambda}{4}$ and $\mu_i = \frac{\mu}{4}$ for *i=0,...,3* and the utilization factor is the same both for the whole and for each subqueue. This allows us to have an analytic comparison among normal tail dropping, R-CP, and R-CS schemes. Since the instantaneous queue size is used in the analysis, it will not be comparable with the results acquired by simulation. However, it will provide an intuition into the subject from an analytical viewpoint. Figure 3.5 depicts the drawing based on equations (3.4), (3.5) and (3.13). It depicts the drop probabilities of an M/M/1/K tail drop queue, an R-CP, and an R-CS queuing management scheme.

In all the schemes a buffer with the size of *K=160* packets is used. In the tail dropping scenario the whole buffer is used for the incoming traffic with a mean arrival rate of λ and a mean service rate of μ. In the R-CS scenario, the instantaneous queue size is used with a *min-th =40*, *max-th = K = 160*, *min-th(i)=10* and *max-th(i)=40* and an incoming traffic with mean arrival rate of λ and mean service rate of μ. For the R-CP scheme, each subqueue has a *min-th(i)=10* and *max-th(i)=40* and an incoming traffic with mean arrival rate of $\lambda/4$, and mean service rate of $\mu/4$.

As expected, the drop probability of a RED queue is higher than the tail dropping queue. This figure also shows that the drop probability of the R-CP sharing scheme is higher than the R-CS scheme. This figure, accompanied with the simulation results presented in Chapter 6, shows the better performance of R-CS.

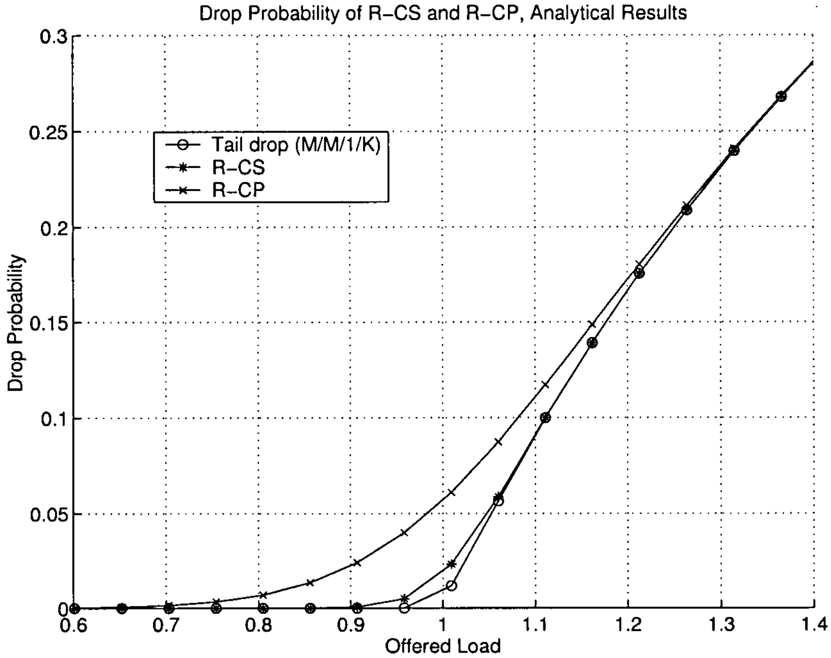

Figure 3.5: Drop Probability of M/M/1/K tail drop, R-CP and R-CS with k=160

3.7 RED Deployment for DiffServ-Based UMTS Backbone Router

In the context of DiffServ, RED can be used for the algorithmic dropper of a DiffServ-based GPRS/UMTS backbone router. It is applicable to those traffic aggregates that utilize TCP for their transport protocol, that is, for interactive and background traffic classes. For the conversational and streaming traffic classes only absolute droppers are used. Thus, an incoming packet with these traffic classes is accepted when it is determined to be conformant and is dropped if it is not. Algorithmic droppers are used for the other 2 classes and, therefore, RED deployment is appropriate. Each of these traffic classes are mapped to some PHB consisting of a set of subgroups.

Due to the simulation results presented in Chapter 6, deployment of RED on shared buffers is proposed for TCP-compatible aggregates. Each traffic aggregate has its own dedicated buffer, but this buffer is shared among the subgroups of that aggregate class and one of RED on shared-memory buffer schemes is used for the algorithmic dropper. This configuration has the advantage of lower packet drop ratio and being reconfigurable, and the disadvantage of slightly higher buffer management complexity.

Chapter 4

DDB-FFQ Scheduling Scheme

This chapter is dedicated to the study of the DiffServ scheduling block. The motivation behind this study is to implement a fair, efficient and simple scheduler that also has decoupled delay-bandwidth guarantees, and supports link sharing objectives. A scheduling block with these properties enables the operator of a DiffServ-based UMTS backbone to define PHBs with different delay and bandwidth guarantees easily. Decoupled Delay-Bandwidth Frame-based Fair Queuing (DDB-FFQ) is a proposed scheduling algorithm that can achieve the above mentioned goals. It is a modification of the Frame-based Fair Queuing (FFQ) scheme, and therefore, inherits all of its properties, while it enables the assignment of decoupled delay-bandwidth guarantees.

This chapter consists of the following sections. First, an overview of scheduling schemes, the link sharing model, and related previous work is provided. Then, the concepts of rate-proportional schedulers, especially the structure of FFQ, are addressed. Finally, a novel scheduling system based on the FFQ, called DDB-FFQ is presented, which has the desirable properties needed for implementation in a DiffServ-based UMTS backbone router. In this chapter, the term "traffic session" is used as an equivalent to DiffServ traffic aggregate.

4.1 Introduction

The quality of service in integrated services packet networks is expressed in terms of delay, bandwidth, jitter and loss ratio. Various applications have different requirements and expect different values of these QoS characteristics. In other words, these parameters might be defined or requested independently of one another, according to the application being used. For example, an application might be very sensitive to delay but require only a small bandwidth, while another application might work well with a loose delay bound but need a large bandwidth and be sensitive to packet losses. Consequently, it is essential to be able to set these parameters independently for each traffic class.

The remarkable module responsible for maintaining QoS is the scheduling element. As such, it has been studied extensively and many scheduling algorithms have been presented in the literature [29][30][31]. Among these schemes, Generalized Processor Sharing (GPS) [32] is purported to be an ideal service discipline in terms of delay and fairness. Although it is based on the fluid model and is not viable for a packet network, its packet-by-packet versions Weighted Fair Queuing (WFQ) and Worst-case Fair Weighted Fair Queuing (WF^2Q) [29] approximate GPS closely. The main drawbacks of these schemes are their complexity and scalability problems.

A series of scheduling policies, called rate proportional servers [30], endeavor to approximate GPS with a time complexity of O(1). In these schemes, the level of service received by session i is set by the value of a parameter called session i rate, r_i. This parameter defines the allocated bandwidth of that session or traffic class. It is also the basis for calculating the delay bound associated with class i, that is, the delay bound of class i is dependent on its allocated rate and can not be defined separately. These schedulers use a number of nondecreasing functions of time called *potentials*. A *system potential* function is used to keep track of the state of the whole system and a *session i potential* is used to keep track of the service session i received during its busy period. These potential functions should be updated continuously to avoid fairness discrepancy.

Frame-based Fair Queuing (FFQ) is a rate-proportional scheduler that recalibrates its system potential periodically with a maximum period called *frame size*. This service discipline is fair, simple to implement, and efficient. However, like other rate-proportional scheduling algorithms, it does not provide decoupled delay bound and bandwidth allocations.

Another important issue related to service discipline algorithms is the concept of link sharing. A link sharing policy is the method of assigning the bandwidth of a link to a set of packet classes. A class can be an aggregate of some traffic streams having the same bandwidth requirements, or the traffic of an organization. The concept of link sharing [33][34] was first introduced mainly for congested periods to restrict services offered to some classes to their link sharing bandwidth. This bandwidth can even be zero, for example, for email traffic stream. In deploying a link sharing policy, there are a number of critical objectives and issues that should be considered as mentioned in [33][35]. First, each class should receive its assigned bandwidth over a suitable interval of time. This requirement should be met both in normal and congestion situations. Second, the excess bandwidth should be distributed to all active sessions according to a rule, not arbitrarily. Excess bandwidth is that bandwidth portion of the link that is unused at a particular time, for example, due to an inactive session. Third, link sharing is essentially a bandwidth oriented method, but delay is as important as bandwidth for many applications. A link sharing implementation should not hinder the general scheduler in providing a delay guarantee to a well-behaved, real-time application. Fourth, link sharing has its highest importance when congestion happens or a traffic aggregate misbehaves. Therefore, it should allocate at least the minimum assigned bandwidth to all classes, including the misbehaving traffic.

Hierarchical Fair Service Curve (H-FSC) [35] is a recently introduced algorithm, which presents a scheduling system with the decoupled delay bandwidth feature. H-FSC is based on link sharing and the Service Curve Earliest Deadline first (SCED) [36][37] scheduling scheme. It is argued in [35] that since there are conflicts between link-sharing and real-time requisites, it is not possible to realize both of them simultaneously at all times, although it is possible to closely approximate the desired model. Since H-FSC is based on service curves, it has implementation complexity for many occasions.

In this chapter, we present a link-sharing real-time scheduler, based on link-sharing and FFQ concepts. The complete scheduling system consists of a number of rate-estimators and a modified FFQ. The novelty of the proposed scheme is that two sets of rates are considered: FFQ-rates and assigned rates. The FFQ-rates are used to determine the delay bound of each class and are a part of the internal state of FFQ, while the assigned rates are used to set the bandwidth received by each class. The rate-estimators measure the rate of the incoming aggregates. As long as the rates of all classes are well-behaved and are within their assigned rates, the scheduler acts like a normal FFQ. However, if the actual estimated rate is above the assigned rate, the scheduler modifies the timestamp of the head-of-line packet of the misbehaved class to restrict it to its assigned rate.

4.2 Frame-based Fair Queuing (FFQ)

Frame-based Fair Queuing [38] belongs to the Rate-Proportional Scheduler (RPS) class [30] and has an $O(1)$ timestamp computation complexity, and performance characteristics close to Generalized Processor Sharing (GPS). There are two types of system complexity related to each service discipline: timestamp computation complexity and priority sorting complexity. The former is the complexity of the algorithm used for calculating the timestamp of each incoming packet and the latter is the complexity of the algorithm used to select the queue with the smallest timestamp. For traffic consisting of N sessions, GPS has $O(N)$ and

FFQ has $O(1)$ timestamp computation complexity, while the sorted-priority complexity of all the scheduling systems depends on the sorting algorithm and is usually in the order of $O(log_2(N))$.

In a rate proportional scheduler, the session *i* receives service depending on its allocated rate r_i. Let $W_i(\tau, t)$ denote the service offered to session i during the interval $(\tau, t]$, then the normalized service received by session *i* is $W_i(\tau, t)/r_i$. The objective of a rate proportional scheduler is to equalize the normalized service of all the busy sessions. An RPS scheduler uses a number of nondecreasing functions of time called *potentials* to keep track of the the actual normalized service received by each session during its busy period. A *potential function* $P(t)$ is defined for the whole system and a *session potential* function $P_i(t)$ is considered for each session, so that for any interval $(\tau, t]$:

$$P_i(t) - P_i(\tau) = \frac{W_i(\tau, t)}{r_i}. \tag{4.1}$$

A system potential should satisfy two properties. First, the system potential must be increased with a rate of at least one, that is, for any interval $(\tau, t]$ that the system is busy:

$$P(t) - P(\tau) \geq (t - \tau). \tag{4.2}$$

Second, the system potential is always less than the potential of any backlogged session. Let *B(t)* denote the set of backlogged sessions at time *t*, then the following becomes true:

$$P(t) \leq P_i(t) \qquad \forall i \in B(t). \tag{4.3}$$

In an RPS system, these potential functions should be continuously updated to enable a close approximation to an actual fluid model in the equivalent GPS system. This recalibration interval is the key feature to associate fairness to the algorithm and is its fundamental difficulty. Different recalibration methods result in different rate-proportional schemes. The session *i* potential should be modified in a way that the service missed by that session during the time that it was not backlogged is also considered.

FFQ uses a frame-based recalibration method, that is, it updates the potential functions periodically with a maximum period of *frame size F*. A *frame period (T)* is the interval of time needed to send *F* bits. The maximum amount of class *i* traffic that can be transmitted during a frame period is defined as ϕ_i and $f_i = r_i/C$ is the fraction of the link rate allocated to session *i*. The relations among these parameters are as follows:

$$\phi_i = f_i * F = r_i * T. \tag{4.4}$$

The ϕ_i is defined such that it is greater than the maximum packet size of class *i*, L_i:

$$L_i \leq \phi_i. \tag{4.5}$$

In FFQ, the delay bound of class *i* is determined by its rate r_i which also defines the portion of bandwidth allocated to that class. For a link with capacity *C* and a scheduler with *N* classes, the total assigned rate should be less that *C*:

$$\sum_{i=1}^{N} r_i \leq C. \tag{4.6}$$

Reference [38] proposes a relaxed method of recalibrating the *potential* functions. Let τ_{k-1} be the last time a frame was updated, then the next update needs to be done when the following two conditions are met:

1) The potential of each active session in the equivalent GPS system belongs to the next frame:

$$P_i(t) \geq kT \qquad \forall i \in B(t) \tag{4.7}$$

where *B(t)* is the set of active classes at time *t*.

2) The potential of each class is less than the beginning of the next frame:

$$P_i(t) < (k+1)T \qquad i = 1, 2, ... \tag{4.8}$$

The implementation of FFQ can be simplified by introducing another function called the *based-potential* function. A based-potential function $S^P(t)$ is a nondecreasing function that has the following properties:

1) $S^P(t)$ is less that the potential of every backlogged session:

$$S^P(t) \leq P_i(t) \qquad \forall i \in B(t). \tag{4.9}$$

2) A finite constant $\Delta P \geq 0$ exists such that the following is true:

$$S^P(t) \geq P_i(t) - \Delta P \qquad \forall i \in B(t). \tag{4.10}$$

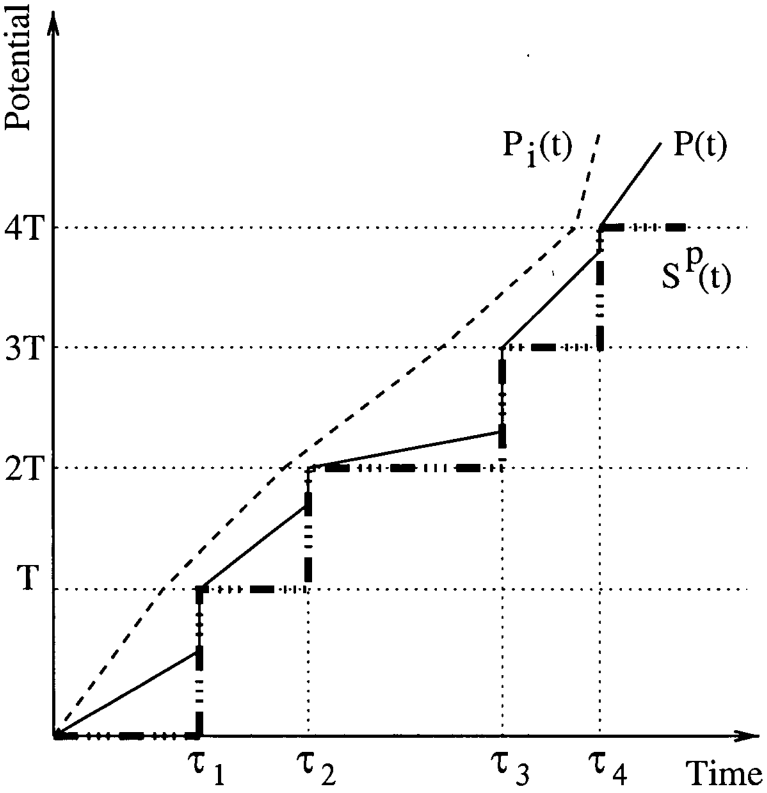

Figure 4.1: Behavior of the base-potential function and potential functions in a fluid FFQ server. $P(t)$ is the potential function, $S^P(t)$ is the base potential and $P_i(t)$ is session *i* potential function. The potential functions are recalibrated at points τ_1, τ_2, τ_3 and τ_4. The frame update points can fall anywhere within a window where each session potential is between *kT* and *(k+1)T [38]*.

Finally it is shown that by using a step function as the system based-potential function and applying the above conditions, FFQ can be easily implemented. Figure 4.1 depicts the relation and behavior of the based-potential and system-potential functions in a fluid FFQ server.

Actually to implement FFQ, [38] shows that it is sufficient to execute the following two algorithms, one at the time of a packet arrival and the other at the time of a packet departure.

The algorithm that needs to be executed on the arrival of a packet in an FFQ server has the following pseudo code:

```
Calculate current value of system potential.
Let t be the current time and $t_s$ the time when the packet
currently in service started its transmission.
    $temp \leftarrow P + (t - t_s)/F$
Calculate the starting potential of the new packet
    $SP(i, k) \leftarrow max(TS(i, k-1), temp)$
Calculate timestamp of packet
    $TS(i, k) \leftarrow SP(i, k) + length(i, k)/\phi_i$
Check if packet crosses a frame boundary
    $n1 \leftarrow int(SP(i, k))$
    $n2 \leftarrow int(TS(i, k))$
    if ( $n1 < n2$) then ( if finishing potential is in next frame)
        $B[n1] \leftarrow B[n1] + 1$ (increment counter)
        mark packet
    end if
```

The algorithm which is executed on the departure of a packet in an FFQ server is this:

```
Increase system potential by the transmission time
of the packet just completed, say j.
    $P \leftarrow P + length(j)/F$
Find timestamp of next packet for transmission
    $TS_{min} \leftarrow min_{i \in B}(TS(i))$
Determine the corresponding frame number.
    $F_{min} \leftarrow int(TS_{min})$
Perform frame update operation if required
    if (packet j was marked) then
        $B[current\text{-}frame] \leftarrow B[current\text{-}frame] - 1$
    end if
    if($B[current\text{-}frame]=0$ and $F_{min} > current\text{-}frame$) then
        $current\text{-}frame \leftarrow current\text{-}frame + 1$
        $P \leftarrow max(current\text{-}frame, P)$
    end if
Store starting time of transmission of next packet in $t_s$
    $t_s \leftarrow$ current time
    Retrieve packet from head of queue and transmit
```

In these algorithms, *P* is the potential state variable, *SP* is the starting potential of a packet, *TS* is the timestamp calculated for each packet, and *B* is an array that is used to keep track of frames and frame boundary crosses. *F* is the frame size, TS_{min} is the smallest *TS* of the head-of-line packets of backlogged classes and *current-frame* is a state variable to keep the value of the current frame. All of these values are reset when the router becomes idle.

4.3 Decoupled Delay-Bandwidth FFQ (DDB-FFQ)

An FFQ scheduler possesses many of the properties expected from a service discipline. It is fair, efficient, and simply implementable. However, it has a crucial shortcoming in that the delay bound of each aggregate class depends on its assigned rate. Therefore, it is not possible to set the delay bound and bandwidth allocation of a session independently. In this section, a novel link sharing service discipline is proposed that overcomes this deficiency. For a system with N traffic classes, the complete scheduling system consists of N rate estimators and a scheduler. The responsibility of each rate estimator is to assess the rate of its corresponding traffic class and to send this estimated rate to the scheduler. The scheduler is a modified FFQ server that uses the statistics gathered by the rate estimator, in addition to its own FFQ states variables, to decide on which queue it should offer service to next.

The novelty of this scheme is that it utilizes two sets of rates for each traffic class: the FFQ-rates and the assigned rates. These two sets of rates are used to realize the decoupled delay and bandwidth property of the system. The FFQ-rates are used to set the delay bound of each traffic class and are equivalent to the r rates defined in a normal FFQ server and the assigned rates are used to set the bandwidth allocation of each aggregate. Each of these sets of rates should obey equation (4.6).

The DDB-FFQ scheduling system's functionality is described as follows. As long as each traffic source is sending packets under or equal to its assigned rate, the scheduler works like a normal FFQ. Since each traffic class is offered service according to its FFQ-rate, its guaranteed delay bound is attained, and because all classes are conformant, they get their assigned portion of bandwidth. Nevertheless, a non-conformant traffic class j is not served with its FFQ-rate. This can be done by adjusting the timestamp of the Head-of-Line (H0L) packet of the queue j when the sorted priority function is performed. It means that the adjusted value is used when the scheduler decides on which queue to serve next, by choosing the queue with the minimum HoL timestamp value, but the packet's original timestamp is preserved. This is done when the second line of the FFQ algorithm for a packet departure (see previous section) is run. This action prevents other classes from service degradation and allows them to get their portion of bandwidth under their delay bounds. While the non-conformant traffic is served at its assigned rate, its delay bound is not met, which is logical in the context of the DiffServ model. There are a number of ways of performing this adjustment. Since this algorithm is meant to be used in a DiffServ-based UMTS router with the best-effort traffic as its lower priority class, the following adjustment is proposed:

$$TS(j) \leftarrow \frac{TS(j) * avg(j)}{min(r)}. \tag{4.11}$$

In this equation, *TS(j)* is the timestamp of the packet at the head-of-line of non-conformant class *j*, *avg(i)* is its estimated rate, and *min(r)* is the smallest assigned rate in the system. This value is used instead of the actual *TS(j)* in the second line of the algorithm for the packet departure.

The value of the weight ω used in the exponential weighted moving average of the rate estimator was chosen as .002. This value is the recommended value for an EWMA system for estimating the rate and was chosen so that it calculates the average rate while it follows the change of stream rate with a reasonable time constant. However, this parameter can be used to set the time constant of the system as desired.

DDB-FFQ can also be described as a link sharing system. In this model, the general scheduler uses a normal FFQ scheme and the link-sharing scheduler is the DDB-FFQ service discipline. In the context of link sharing, a general scheduler is used during normal operating conditions of the network, and the link-sharing scheduler is used during congestion. When congested or faced with a non-conformant traffic, the offered service rate of that session is limited to its allocated rate, and therefore, the delay bound guarantee is not met. In this system, the excess bandwidth is distributed among all the active classes according to their FFQ-rates, and when congestion happens or a misbehaved class exists, it is distributed according to the method used for the time stamp adjustment. In our case, it is distributed according to the adjustment shown in equation (4.11), that is, among well-behaving classes according to their FFQ-rates and among misbehaving classes proportional to the smallest FFQ-rate in the system.

In a DDB-FFQ server, the delay bound observed by an aggregate is determined by its assigned FFQ-rate. Therefore, the proper setting of the FFQ-rate value is the key factor that leads to a favorable delay bound. It is shown that for an FFQ scheduling scheme the delay of session *i* is bounded according to the following equation [30]:

$$D_{max}(i) \leq \frac{L_i}{r_i} + \frac{L_{max}}{R} \tag{4.12}$$

where L_i is the maximum packet size of the session *i*, r_i is the FFQ rate assigned to session *i*, L_{max} is the maximum packet size of all sessions and *R* is the output service rate of the scheduler. Considering its inheritance from the FFQ scheme, this equation can be directly applied to a DDB-FFQ system as well.

This equation illustrates some important facts. First, the delay bound is inversely proportional to FFQ-rate. Second, the delay bound of session *i* is dependent on the maximum packet size of session *i* traffic. Third, it also depends on the maximum packet size of all sessions, even of the session with the lowest priority. This is a logical result of the fact that DDB-FFQ is a work-conserving non-preemptive service discipline.

In the context of a DiffServ-based UMTS backbone network, the delay bound dependency on packet size can be easily overcome by the IP tunneling capability of the UMTS backbone network. At the time of packet encapsulation, the ingress edge router performs packet segmentation to limit the maximum packet sizes. The segmented packet can be reassembled later at the egress edge router. By having the maximum packet sizes and the FFQ-rate values of the DDB-FFQ server, the router operator can set the delay bound of every PHB according to its requirements.

4.4 The Rate Estimator Module/Meter

One of the building blocks of the proposed link sharing, real-time, scheduling system is the rate-estimator module. This element determines the rate of each class, which is the basic statistic fed to the DDB-FFQ scheduler. DDB-FFQ uses this data to manage the bandwidth allocation of each session and to provide its link sharing capability. In the context of a DiffServ-based UMTS backbone router, this module can serve additional purposes, that is, it also acts as the metering element for the purpose of traffic shaping, remarking and/or for billing purposes.

There are several ways of implementing a rate estimator. The usual algorithm used for estimating the traffic rate is the exponential weighted moving average (EWMA) lowpass system [33] which is also the method that is used in the proposed DDB-FFQ scheme. The key parameters of this estimator are its time constant and the frequency at which the rate estimation is performed.

The rate estimator used in DDB-FFQ updates the value of the estimated rate at the time of each packet arrival. The algorithm actually first estimates the packet inter-arrival time. It calculates the difference between the actual and allocated inter-arrival time of each packet and then finds its EWMA values. By having the estimated packet interarrival time, its reciprocal, the packet rate is determined. Assume t is the interval between the time when the last packet has arrived and the time that the most recent packet has arrived. Also assume S is the size of the arrived packet, b is the bandwidth allocated to this packet class, and τ is the allocated interarrival time of this class. Then the discrepancy between allocated and actual inter-arrival times is this:

$$diff = t - \frac{S}{b}. \tag{4.13}$$

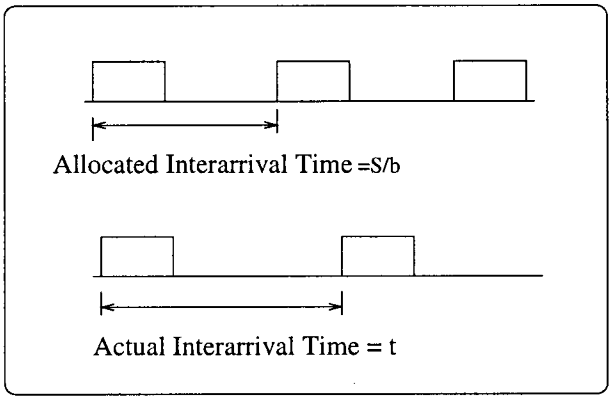
Allocated Interarrival Time =S/b
Actual Interarrival Time = t

Figure 4.2: Variables Used for interarrival time estimation.

Then the exponential weighted moving average (EWMA) model is used to calculate the average of *diff* as in the following:

$$avg_{diff} \leftarrow avg_{diff} * (1 - \omega) + diff * \omega \tag{4.14}$$

where ω is the weight factor and is the key factor for setting the EWMA's time constant. By applying avg_{diff} in place of *diff* in equation (4.13) and considering $t = S/r$, we have the following:

$$avg_{diff} = t - \frac{S}{b} = \frac{S}{r} - \frac{S}{b} \tag{4.15}$$

where *r* is the estimated rate. Finally the estimated rate is simply calculated this way:

$$rate \leftarrow \frac{1}{\frac{1}{b} + \frac{avg_{diff}}{S}}. \tag{4.16}$$

The weight ω denotes the time constant of the estimator. The time constant is the time that it takes for the system to change from its original value to 63% of it. Therefore, to calculate the time constant of this estimator, we first calculate the number of packets it takes for the value of avg_{diff} to move 63% away from its initial value. Suppose *diff* changes from 0 to *I* and let $\omega * I = c$ and avg_{diff} at the time of packet *k* arrival represented by Y_k. Then equation (4.14) can be rewritten in the form of a difference equation:

$$Y_k = (1 - w)Y_{k-1} + c, \qquad Y_0 = 0. \tag{4.17}$$

The solution to this equation is simply given as : $Y_k = (1-w)^k(Y_0 - c/w) + c/w$. Therefore, *k*, the number of packets required for *Y* to become $Y_k = 0.63c/w$ is this:

$$\frac{-1}{ln(1-\omega)}. \tag{4.18}$$

This means that if the rate changes, and therefore causes the *diff* variable to change, it takes -1/ln(1-ω) packets before the value of *avg* moves 63% from its initial value. Since the packet rate can be assumed to be s/b, the approximate time constant is calculated thus:

$$\frac{-s}{b * ln(1-\omega)}. \tag{4.19}$$

Chapter 5

Mapping and Classification

5.1 Introduction

Mapping and classification are two related functions that a DiffServ node must perform to enable proper service differentiation. The mapping function is the act of translating the QoS parameters of one system to another QoS system, in order to maintain a consistent level of service received by a packet along its transmission path, from its source to its destination. In UMTS, the QoS required is identified by a number of attributes associated with each session. DiffServ uses its own method of differentiating between distinct aggregates, marking the IP header of their packets and handling them with different PHBs. When DiffServ is deployed in the UMTS backbone network, the QoS parameters of a packet passing the boundary of the core network should be mapped to the QoS constraints of the new system in the edge router. If an incoming packet originates from an MS into the core network, its UMTS QoS attributes should be mapped to an equivalent DSCP, which is pre-defined in the DiffServ-aware core network. If it is outgoing from a core network, originating from an external source, the PHB should then be translated to equivalent UMTS QoS parameters. Mapping might also be needed between the external bearer service QoS system and DiffServ. This chapter is focused mainly on the mapping function from the UMTS QoS system to DiffServ. Classification, on the other hand, is the act of splitting an incoming traffic into a number of output streams, by using filters. In DiffServ, the traffic is simply classified by applying match conditions on the DSCP field of the packet header.

This chapter first studies the QoS mechanism of UMTS/GPRS and that of DiffServ. Then the function of mapping these QoS systems to each other is addressed, and finally, the classification methods are presented.

5.2 UMTS/GPRS QoS Classes and Attributes

In UMTS/GPRS, Quality of Service (QoS) is conveyed by a QoS profile. A QoS profile is a part of the PDP context which is negotiated with the user to acquire his demanded grade of service. It is a single parameter with multiple attributes. Each of its attributes defines a specific characteristic of the traffic. These attributes as defined in the specification are *traffic class, maximum bitrate, guaranteed bitrate, delivery order, maximum SDU size, SDU format information, SDU error ratio, residual bit error ratio, delivery of erroneous SDUs, transfer delay, traffic handling priority, and allocation/retention priority* [2]. Hereafter, some of these attributes are explained.

Traffic Class: The most noteworthy attribute is the traffic class. In defining the traffic classes, restrictions and limitations of the air interface have been taken into consideration to develop reasonable service provisioning, while avoiding complex mechanisms. There are four UMTS traffic classes: *conversational class, streaming class, interactive class* and *background class*, listed according to their relative priority, from highest to lowest. Conversational and streaming classes are meant to carry real-time traffic streams. The main distinctions between conversational and streaming classes are their delay and delay variation sensitivities. Interactive and background classes, however, are intended to cover all traditional

Internet applications, such as WWW, email and FTP.

Conversational Class: This class is used for conversational and multimedia applications, such as GSM telephony speech, voice over IP and video conferencing. These applications are mainly used by two (or more) peers of human end-users. The fundamental QoS characteristics of this class are these: preservation of the time variation between consecutive entities of the traffic, and stringent low delay.

Streaming Class: This class carries the traffic of a video or audio stream from a host machine to a human end-user, and is basically a one way transport. Its QoS characteristic is that it needs that the time relation between the traffic consecutive entities be preserved. However it has a much lower delay sensitivity compared to the conversational class. An example of its application is MP3 video streaming or Internet radio.

Interactive Class: This class is for applications where an end-user (human or machine) is on-line, asking some data from a host machine. The main characteristic of this class is its request-response pattern, with a lower delay sensitivity compared to the above traffic classes. It may only need payload content preservation.

Background Class: When an end-user (mainly a machine) sends or receives data in the background without any delay sensitivity, this class is applied. Examples of this kind of application are email and database downloading.

Maximum Bitrate (kbps): This attribute defines the maximum rate that can be delivered to or by UMTS. Traffic is conformant with a defined maximum bitrate when shaped by a token bucket with a token rate of the maximum bitrate, and a bucket size of the maximum SDU size.

Guaranteed Bitrate (kbps): It defines the guaranteed rate, that is, the guaranteed number of bits within a period of time, divided by the duration of that period, delivered by UMTS. QoS requirements, such as delay and reliability are only guaranteed for traffic rates up to this guaranteed bitrate.

Delivery Order: This attribute specifies an application requirement, whether its packets need to be delivered in order or not.

Maximum SDU Size: It defines the maximum allowable SDU size.

SDU Format Information: This attribute defines the list of possible *exact* sizes of SDU. If an application is able to state its SDU sizes, the bearer services are less expensive, because this data can be used by UTRAN at the RLC layer.

SDU Error Ratio: It indicates the ratio of dropped or erroneous packets of a conforming traffic.

Transfer Delay (ms): This attribute indicates the maximum delay in the 95^{th} percentile of the distribution of delay observed when a SDU is carried from one Service Access Point (SAP) to another SAP.

Delivery of Erroneous SDUs: It indicates whether an erroneous SDU shall be delivered or discarded.

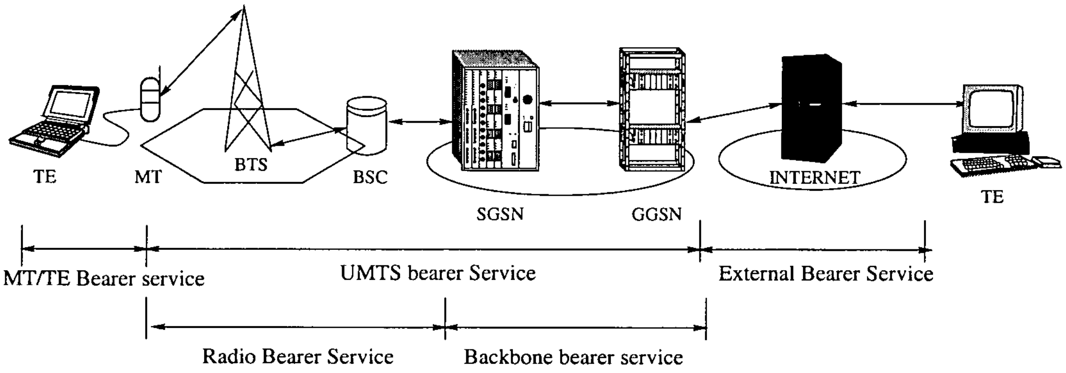

Figure 5.1: A schematic of the structure of the GPRS backbone network

Table 5.1 shows the range of the values expected of a UMTS bearer service QoS attributes.

Traffic Class	Conversational Class	Streaming Class	Interactive Class	Background Class
Max bitrate (kbps)	2048	2048	2048	2048
Delivery order	Yes/No	Yes/No	Yes/No	Yes/No
Max SDU size (octets)	1500 or 1502	1500 or 1502	1500 or 1502	1500 or 1502
Errorneos SDU delivery	yes/no	yes/no	yes/no	yes/no
Residual BER	5e-2 to 1e-6	5e-2 to 1e-6	4e-3 to 6e-8	4e-3 to 6e-8
SDU error rate	1e-2 to 1e-5	1e-1 to 1e-5	1e-3 to 1e-5	1e-3 to 1e-5
Transfer delay(ms)	max 100	max 250		
Guaranteed bit rate	2048	2048		

Table 5.1: Expected range for UMTS bearer service QoS attributes

5.3 DiffServ PHB Groups

The DiffServ working Group of IETF has defined a number of different PHB groups for different applications. In this section an overview of all the up-to-date PHBs are presented.

Best Effort (BE) PHB Group: BE is the traditional Internet traffic class which has been used from the beginning. This PHB implies that there are some network resources available and there is a good faith commitment that the nodes in the path will do their best to forward the packet in a fair manner, but there is no guarantee of its delivery or of its received level of service. The recommended DSCP for this PHB is 000000.

Expedited Forwarding (EF) PHB Group [39][40]: EF PHB is aimed at providing a low loss, low latency, low jitter, assured bandwidth edge-to-edge service. It has also been called Premium Service. A codepoint of 101110 is recommended as its DSCP, and therefore, there can be only one instance of EF in a DS domain if one follows the recommended DSCP.

Assured Forwarding (AF) PHB Group [41]: This group provides four independently forwarded AF classes. A packet in each of these AF groups can be assigned to three dropping precedences. Table 5.2 summarizes the recommended DSCP for these twelve instances of AF.

The value of each class codepoint can be classified as a six bits binary, with the form of ‘cccddd’, in which ‘ccc’ defines the AF’s class and ‘ddd’ indicates its drop precedence.

Drop Prec.	Class 1	Class 2	Class 3	Class 4
Low	001010	010010	011010	100010
Medium	001100	010100	011100	100100
High	001110	010110	011110	100110

Table 5.2: Recommended DSCP values for AF PHB

Network Control (NC) PHB Group: An NC packet is used for carrying any possible signaling data needed in the DiffServ control plan to exchange management information between DiffServ routers within a DiffServ domain. Its recommended codepoint is 11x000.

Lower than Best Effort (LBE) PHB Group [42]: The primary goal of defining this PHB group is to provide an alternative to best effort traffic at the time of congestion. This PHB carried packet of an application that has a lower drop precedence, compared to best effort traffic.

Dynamic Real-time/non-real-time (RT/NRT) PHB Group [43]: This group provides delivery of real-time and non-real-time traffic.

Bulk Handling (BH) Per Domain Behavior (PDB) Group [44]: BH PHB is used for traffic that may be starved even in a properly functional network. This starvation is not a requirement of the PHB, but is used for comparison with the best effort PHB. It can be used for carrying packets of an application that has sufficiently low value.

Alternative Best Effort (ABE) PHB Group [45]: This PHB group gives the best effort applications an option: to choose between receiving lower delay or receiving higher throughput by defining two types of green and blue packets.

Assured Rate (AR) PDB Group [46]: This PDB defines the overall treatment a packet receives throughout a DiffServ domain. This PDB is suitable for carrying the traffic of applications that need rate assurance but do not require delay bounds.

5.4 Mapping

The mapping function is the first and the most important step in supporting the QoS of a UMTS stream. This is the place where a PHB suitable for a session is considered and a DSCP is selected for it accordingly. The selected PHB defines the level of service that stream will observe throughout that DS domain.

The mapping function is an implementation dependent task, however. It depends on the number of different QoS provided and the level of QoS fineness. In this section a basic mapping system is presented. A simpler but completely analogous mapping method has been used in our simulation model.

A close observation of the standard PHB suggested by IETF shows that there are not enough PHBs to cover a fine set of PHB equivalent to the UMTS QoS system. Nevertheless, a basic methodology can be formed. Motivated by the UMTS traffic classes, we classify the traffic types into five PHB groups, *G0* to *G4*:

G0: This group is used for special, infrequent network management control dedicated to the DiffServ control plane for the purpose of signaling between network elements. This Group has a very high priority over other groups. A DSCP with the value of 11x000 has been assigned to the NC PHB, thus there can be two subclasses by using the recommended DSCP.

G1: This group is used as an aggregate of traffic streams that have stringent delay sensitivity, for example, the conversational traffic class in UMTS. The proper PHB for this group is EF PHB. Nevertheless, only one DSCP value, 101110, was advised by IETF. No other subgroups can be defined if one wants to be consistent with the standard. However, this problem can be easily overcome by using some other DSCPs, as allowed in IETF specifications, or by mapping subgroup one to EF PHB and mapping the rest of the subgroups to the higher priority AF PHBs. These two choices only affect the value of DSCP chosen, but do not affect the overall performance of the system, as far as the proper PHB policies are selected.

G2, G3: Group two is aimed at aggregating streams with the streaming traffic class and Group three is for the interactive traffic class. The most rational PHBs that can be used for these groups is a set of Assured Forwarding PHB groups. This provides a total of up to twelve subgroups.

G4: This group is used for aggregating non-delay sensitive traffic. It is equivalent to the UMTS background traffic class. This group can have a number of subgroups and can be mapped to these proposed PHBs: best effort PHB, lower that best effort PHB, alternative best effort PHB and bulk handling PHB.

5.5 Classification

In the DiffServ context, a Classifier is a functional block that looks at the DSCP value marked on each IP packet's header to split an incoming traffic into N output streams. An important characteristic of a classifier is that it should work unambiguously, and classify uniquely the incoming traffic.

A classifier is usually implemented by using filters, and is identified by its set of filters. A filter is a DiffServ block that applies a set of conditions to the value of a classification key, such as the packet header, content, or attributes to categorize the packet into a finite set of groups. In a behavior aggregate (BA) DiffServ classifier, the classification key is the packet's DSCP field. In a BA classifier, a filter is defined by a value that is matched with the DSCP of the incoming packet to determine its logical output stream. This value can be a six bits binary number, like 111000, and can be applied with different match conditions, such as wildcard, prefix, masked, range and/or exact match.

A Multi-Filed (MF) classifier classifies the packets based on two or more fields in a packet header. The most usual header fields used are destination address, source address and DSCP in the IP header, and source port and destination port in the TCP header.

For a DiffServ-based UMTS backbone router, a packet's PHB can be determined just by the value in its DSCP field in the IP header. Therefore, a simple BA classifier is sufficient, which uses exact match conditions for conversational, streaming and interactive traffic groups. An exact condition match with a filter value of 000000 is used for the best effort subgroup of the background traffic group and a wildcard (******) match is used to put any packet with an unknown or ambiguous DSCP value into an output stream with lower priority, such as lower than best effort or bulk handling PHBs.

Chapter 6

Simulation Model and Results

This chapter addresses the simulation model and results, presenting the discussion in three parts. The first part presents simulation results comparing the performance of the RED on shared buffer schemes, as discussed in Chapter 3, under both TCP and non-TCP traffic models. The second part presents simulation results related to the scheduling system, as discussed in Chapter 4, by providing proof of the decoupled delay bandwidth capability of DDB-FFQ and its link sharing ability. While parts one and two evaluate the scheduling and dropping elements of a complex DiffServ system individually in a single node model, part three presents the simulation model of a simple DiffServ-based UMTS backbone network with three nodes and a more realistic traffic model. This model utilizes a more complete configuration of a DiffServ router to add QoS support. The scheduling block of each router is based on the DDB-FFQ service discipline.

The OPtimum NETwork performance (OPNET) modeler is the simulation environment used for building and simulating these models. OPNET is a renowned, widely accepted, industry oriented software that enables researchers to model and simulate complex network scenarios, communication protocols and traffic models. It is based on a hierarchical structure that allows unlimited subnetwork nesting. While OPNET has a fair number of pre-defined models, it allows building new models by using finite state machine modeling at it lowest level, which can be programmed by Proto-C language. Proto-C is a combination of general C/C++ facilities and an extensive library of built-in, high-level commands, known as Kernel Procedures.

OPNET has an easy-to-use graphical user interface and a variety of tools for specification, data collection, simulation and analysis, and presentation of results. The package consists of several editors, each focusing on a particular aspect of modeling. It is based on an essentially hierarchical structure for modeling actual networks or distributed systems. There are three major modeling domains: the network domain, the node domain, and the process domain.

For example, for modeling a simple office LAN, first the actual network is modeled in the network editor using a few nodes and links. Each of these nodes is defined in a node model consisting of a number of modules. Finally, the functionality of each of these modules is described in a process model, using state machines and Proto-C programming.

Data collection is performed in the form of three types of output. These are output vectors, output scalars and animation. An output vector is a collection of pairs of real values and usually presents the system's performance versus simulation time. In contrast, output scalars are used to collect individual data points across several instances of simulation. Animation is rarely used, but it provides a powerful visual tool for debugging.

6.1 Dropper Element

This section compares the characteristic of different RED on shared-memory buffer algorithms that were presented in Chapter 3. Although RED is basically used to interact with an end-to-end congestion manager, such as the TCP congestion control and avoidance mechanism, both TCP and non-TCP traffic sources should be studied for its deployment. The

Endpoint Congestion Management Working Group of the IETF intends to integrate congestion management across all applications and transport protocols by introducing a universal congestion manager (CM)[18]. This RFC proposes an integrated congestion management universal among all applications and transport protocols that enables application adaptation to congestion, in addition to providing efficient multiplexing. However, before the required implementation of the CM module, TCP will be the only transport protocol that utilizes an end-to-end congestion manager. Still, many other applications use UDP and their congestion avoidance algorithms are inadequate or non-existent. Almost all real-time and streaming applications are UDP-based and non-responsive to congestion notification. Therefore, it is convenient to classify the traffic streams into two groups: TCP-compatible flows and non-responsive flows.

This section consists of two discussions, regarding the above traffic categorizations, one for non-TCP and one for TCP traffic flows. In the former, the traffic handling capability of the router in terms of packet loss and queuing delay is evaluated for instances when early packet dropping has no meaning to the source. In the latter, the performance of the algorithms in avoiding congestion with maximum usage of network capacity is tested for instances when the traffic source can sense traffic conditions.

The simulation results are used to evaluate and compare the following properties of the schemes:

- the loss ratio seen when traffic sources do not use an end-to-end congestion manger;
- the loss ratio and utilization when TCP is the transport protocol of the traffic used, in addition to the throughput of the system;
- the packet delay.

Although the main focus here is to compare the performance characteristics of RED algorithms on a shared-memory buffer, RED on dedicated and shared memory buffers are also compared, because of the identical structures of an R-CP scheme and RED on a dedicated memory queues.

6.1.1 Simulation Model

Although two different traffic models, TCP-based and non-responsive, have been used, the simulation model itself has a single structure. Figure 6.1 shows the schematic of the simulation model. It consists of four traffic sources, a gateway with a shared-memory buffer and RED deployment, and the traffic destination. Four traffic generators provide four traffic streams which are multiplexed into a single flow by the Mux module.

The gateway itself is composed of a classifier element that separates the incoming traffic into four classes, a shared memory buffer with RED deployed on it and a scheduler. The traffic patterns of the traffic generators are described later in each section. The scheduler is a simple work-conserving, non-preemptive, round-robin scheduler. A simple scheduler has been deployed to allow the evaluation to reflect only the characteristics of the dropping algorithms. To compare the three congestion management schemes, three scenarios have been evaluated. These scenarios differ only in their RED algorithms, but traffic patterns, simulation parameters, and other parts are similar, making the comparison possible.

6.1.2 Performance of RED on Shared Memory Buffer Without End-to-End Congestion Control

For a non-responsive traffic source, the packet dropping at the router has no meaning in terms of congestion notification. Still the RED dropping has some of its advantages, such as having a lower average queuing delay and lower bias against bursty traffic.

In this model each of the traffic sources (Figure 6.1) generates packets with exponentially distributed interarrival time and service duration. The RED parameters for each

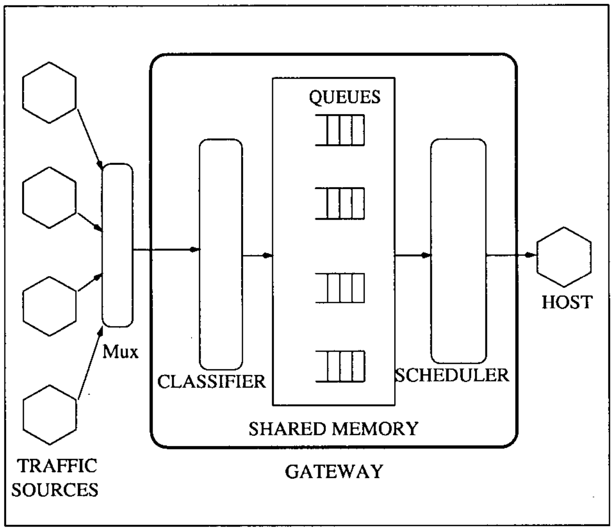

Figure 6.1: The schematic of the simulation model used to evaluate the algorithmic droppers

subqueue are chosen as follows [47]:

min-th(i) = 5 * average packet size

max-th(i) = 3 * min-th

weight factor = 0.002

Max drop probability = 0.1

The values of the maximum and minimum thresholds for the whole queue (used in R-CS and R-SMA) are calculated thus:

$max\text{-}th = \sum_{i=0}^{3} max\text{-}th(i)$

$min\text{-}th = \sum_{i=0}^{3} min\text{-}th(i)$

Two types of traffic patterns have been used to allow performance evaluation of these schemes under different conditions. These patterns are called cases A and B.

Case A: in this case, all sources are generating symmetric traffic, that is, each has an average base traffic rate of 9600 bps and the gateway has a service rate capability of 38400*1.03 bps at all times where the 1.03 factor is used to ensure stability. Both packet interarrival times and packet lengths are generated using exponential distributions. In this case the performance of the system is evaluated when all classes are generating packets with overloaded rates of 100% to 400%. Therefore, every class suffers the same amount of overload and the router is congested with the same share of each class. This traffic pattern was chosen to demonstrate the performance comparison of RED schemes under the same congestion conditions. The physical queue size of the buffer in each scenario has been set to infinity to allow evaluation of the packet dropping due to RED algorithm only. Still, the queues do not grow continuously because of the maximum threshold used in RED algorithms. Table 6.1 shows the value of RED parameters used in the model.

min-th (bits)	max-th (bits)	Buffer size	Weight factor	Max Drop Probability
48000	114000	infinite	0.002	0.1

Table 6.1: RED Parameters

Figure 6.2 shows the packet drop ratio of the three schemes when their load is increased from 100% to 400% of their base rates. The packet ratio is the number of packets dropped over the total number of packets generated during the simulation. Due to RED algorithm, the schemes initiate packet dropping at approximately 75% of the normal rate, uniformly. However, beyond a 150% load, they show distinct packet losses. This figure shows that when traffic is symmetric, the R-CS has the least, and the R-CP has the worst packet dropping rates.

Figure 6.3 shows the EWMA queue size of the buffer of each of the schemes. This is not indicative of the average delay of each queue, but shows the overall status of the buffers, as it is the average value that RED algorithms calculate to decide on whether a gateway is congested or not. This figure also shows that R-CS has a lower value and indicates that it handles traffic more smoothly, without issuing a premature congestion notification.

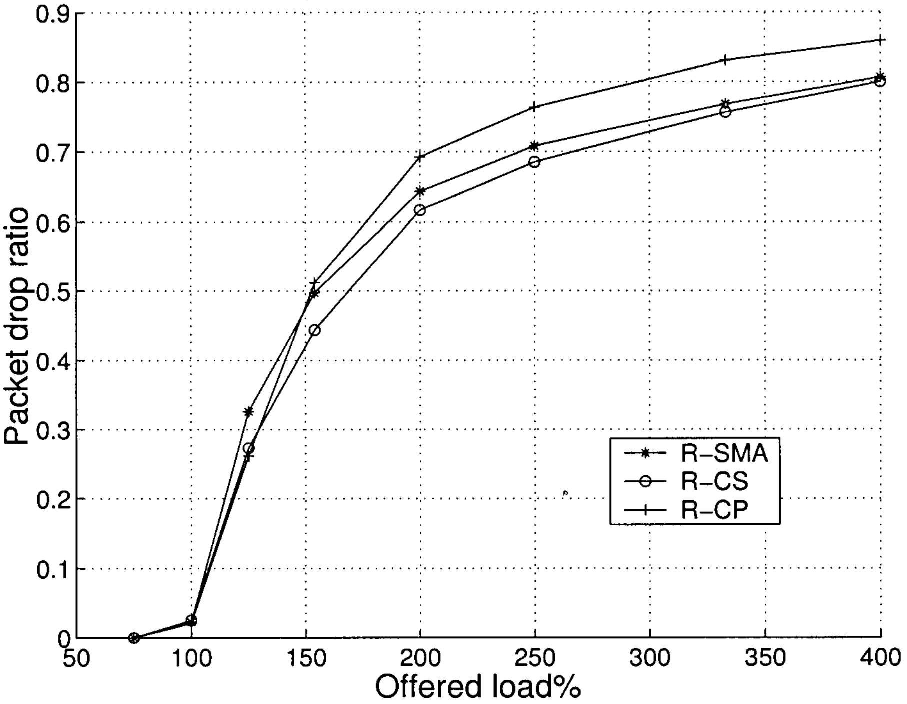

Figure 6.2: Packet drop ratio

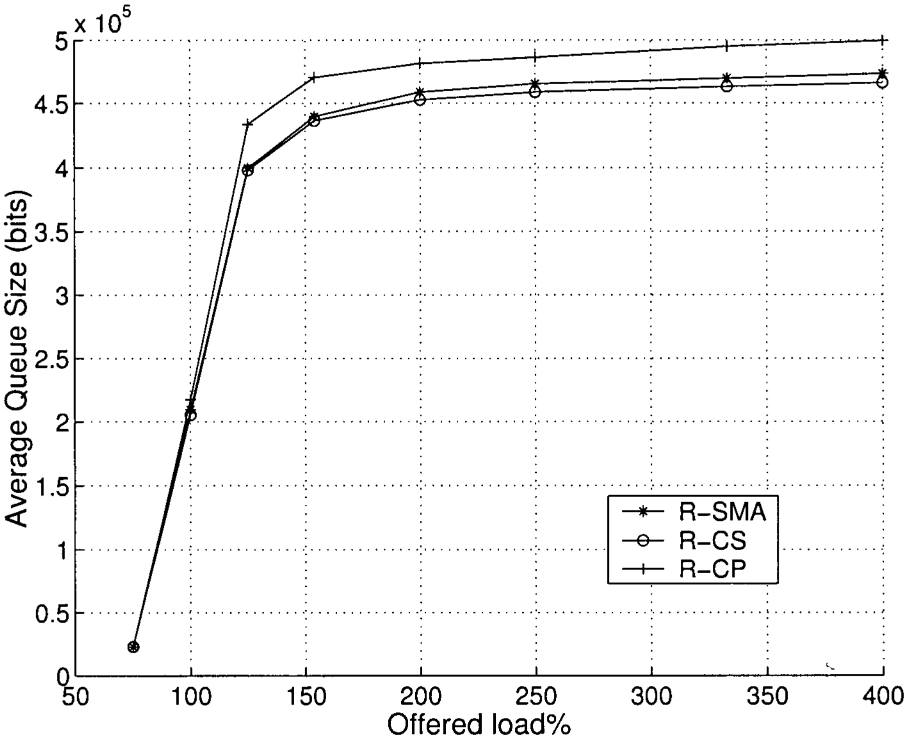

Figure 6.3: Average queue size

Case B: In this case, the characteristics of the schemes under a different traffic pattern are evaluated. Here, only two out of the four traffic generators, class 1 and class 3, cause congestion at the router. These sources overload the router in four intervals during the simulation. In a simulation with a duration of one hour, they constantly generate packets with their base rates, with an added traffic load of 150% to 300% during 10 to 14, and 35 to 39 minutes for class 1 and during 20 to 24, and 50 to 54 minutes for class 3. The simulation RED parameters are set according to table 6.2:

min-th(i)	max-th(i)	Rate of class 0 & 2	Base rate of class 1 & 3	Buffer size
48000 bits	600000 bits	8000 (bps)	9600 (bps)	1.25 * max-th

Table 6.2: RED Parameters

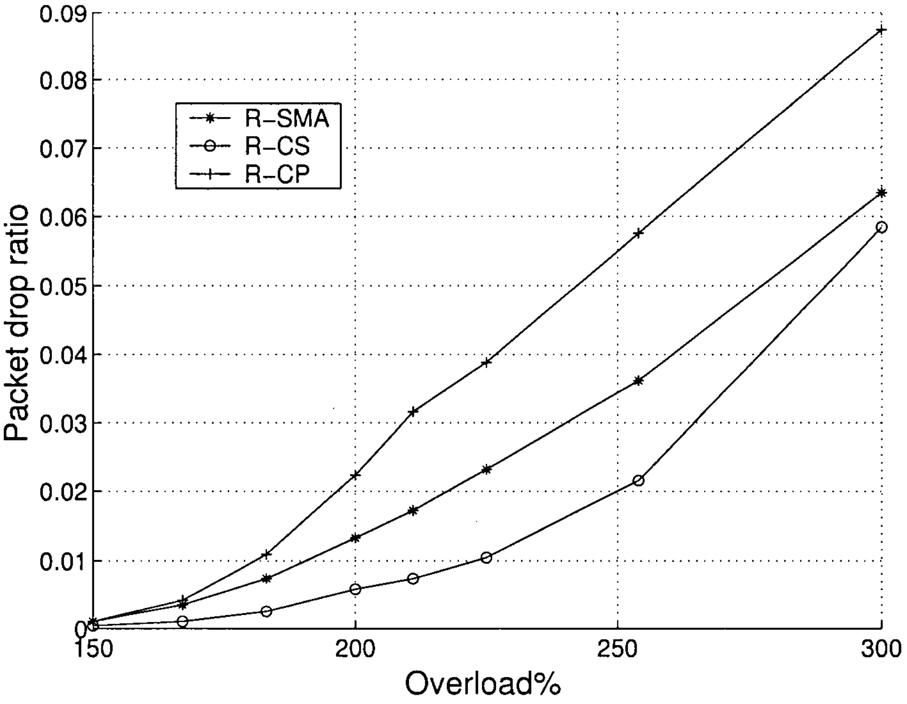

Figure 6.4: Packet drop ratio

Figure 6.4 shows the packet drop ratio of R-CP, R-CS and R-SMA under this new traffic pattern. The horizontal axis, overload%, is the percentage of the added load for the

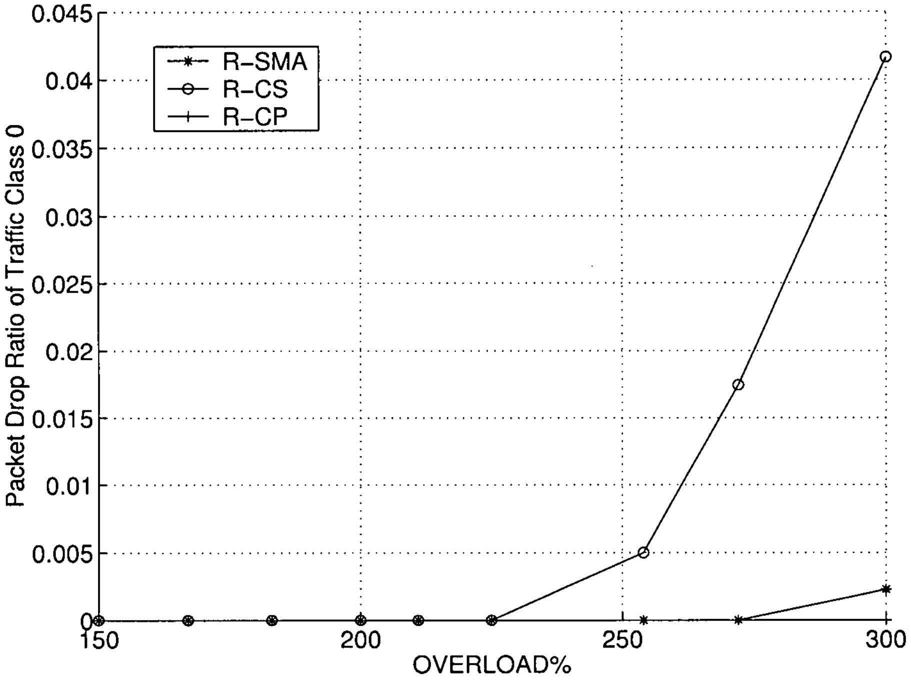

Figure 6.5: Packet drop ratio class 0

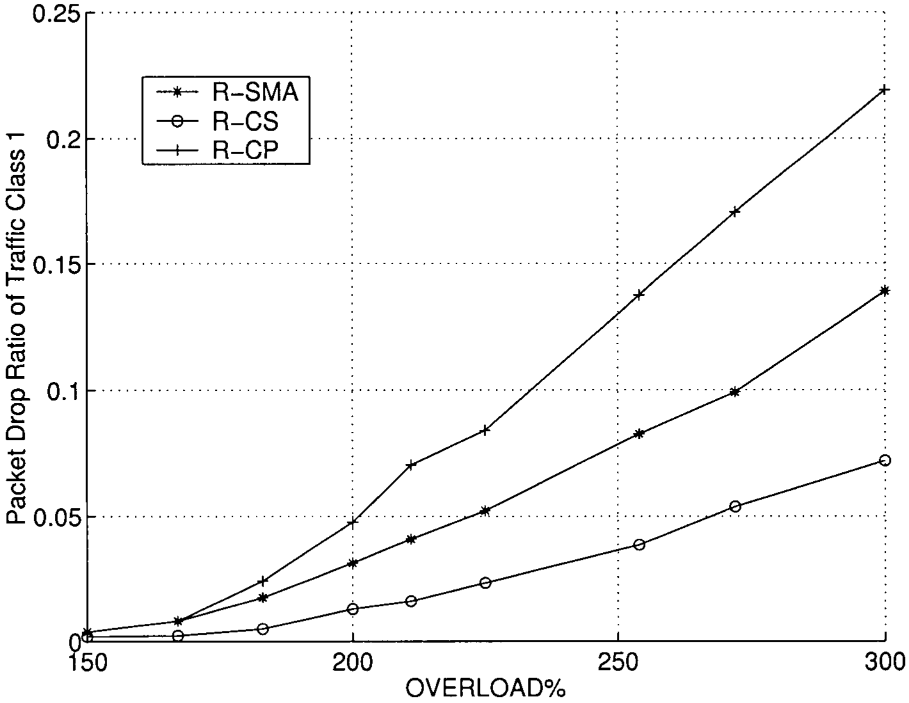

Figure 6.6: Packet drop ratio class 1

traffic class 1 (or 3). This figure shows the same results as Figure 6.2 with a more distinct performance difference. It shows that R-CS has the smallest and R-CP the largest overall packet dropping rate. This is due to the fact that in R-CS, the unused space of other traffic aggregates can be shared. However, the smaller total packet dropping of R-CS has the price tag of larger packet dropping rates for classes 0 and 2, as seen in figure 6.5 for class 0. Nevertheless, this higher packet dropping happens in the high values of overload and has a relatively small drop ratio. Figure 6.6 demonstrates the packet dropping ratio of traffic class 1. Because of their analogous traffic pattern, the performance of class 2 is similar to that of class 0 and class 3 is similar to class 1.

Figures 6.4, 6.5 and 6.6 together depict that using R-SMA under the mentioned traffic pattern improves the packet dropping rates in the overall buffer and of classes 1 and 3 in the worst case load scenario, with only a small increase in packet dropping of well-behaved traffic.

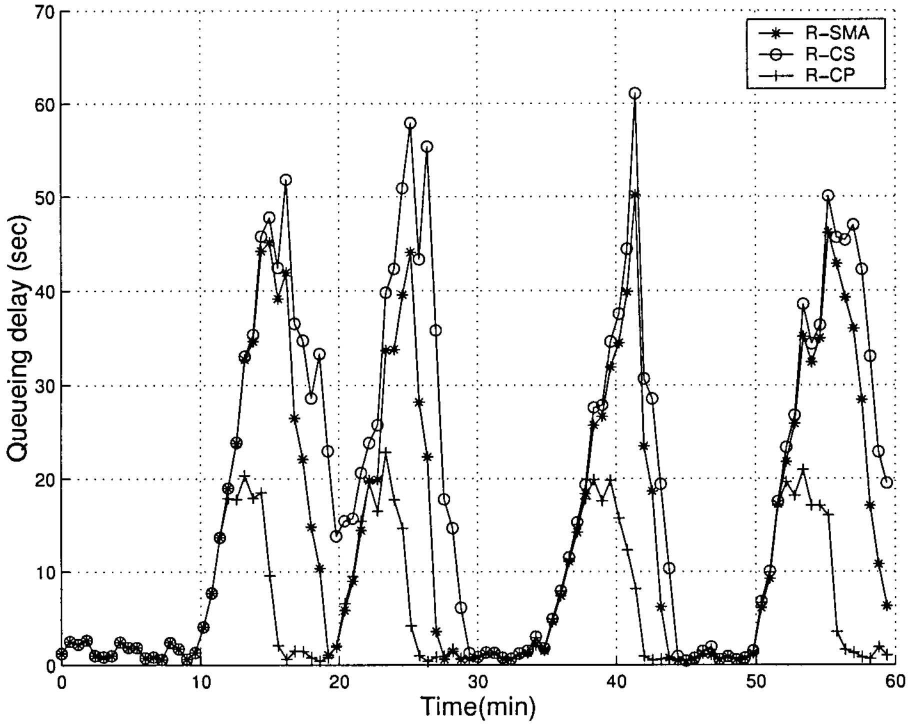

Figure 6.7: Queuing delay

To demonstrate the time based behavior of the schemes, the simulation results for a model with the same traffic pattern as in the above, and with a class 0 and a class 3 added overload of 175%, were sampled.

Figure 6.7 shows the queuing delay of R-CS, R-CP and R-SMA schemes for a simulation duration of one hour within the traffic pattern of case B. As expected, less packet dropping of R-CS will yield to a higher delay, while a higher drop rate of R-CP leads to a lower delay at a time of congestion. However, these schemes are proposed to be used for interactive and background traffic classes, which have a higher priority for content preservation than packet delay. In addition, the maximum queuing delay can be controlled by the queue size. This figure also clearly shows the traffic pattern. As expected, the congestion has occurred in the intervals of [10, 14], [20, 24], [35, 39] and [50, 54]. Since the R-CP scheme has lower packet drops, it has more backlogged packets and it takes longer to return to its normal queuing delay.

Figures 6.8, 6.9 and 6.10 demonstrate the dropping pattern of the schemes during a sampled simulation. These figures demonstrate two facts. First, packet dropping ratios in R-CP and R-SMA are more spread out than in R-CS, while packet dropping in R-CS mainly happens during the peak queue congestion periods. Second, the maximum value of the drop ratio is low for R-CS, and high for R-CP. R-SMA has a moderate maximum drop ratio. This maximum value is an indicator of burst packet dropping. Therefore, one can conclude that R-CS handles bursty traffic more efficiently.

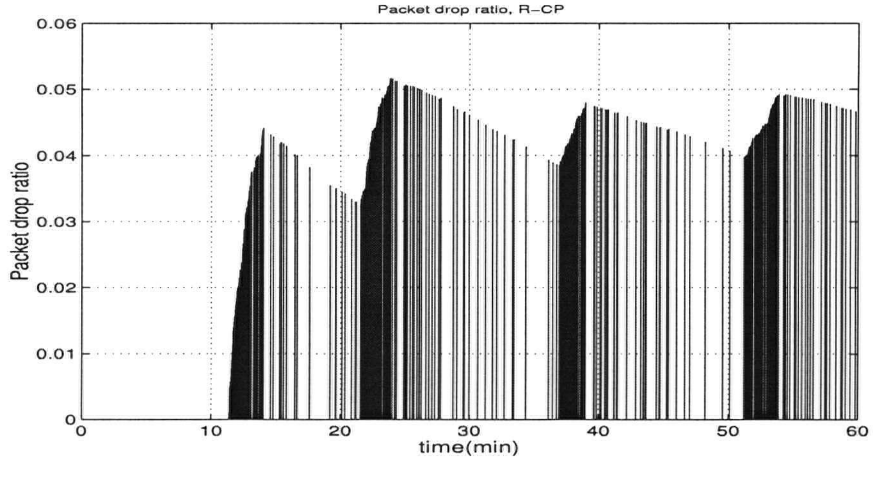

Figure 6.8: Packet drop pattern of R-CP, Load%=175%

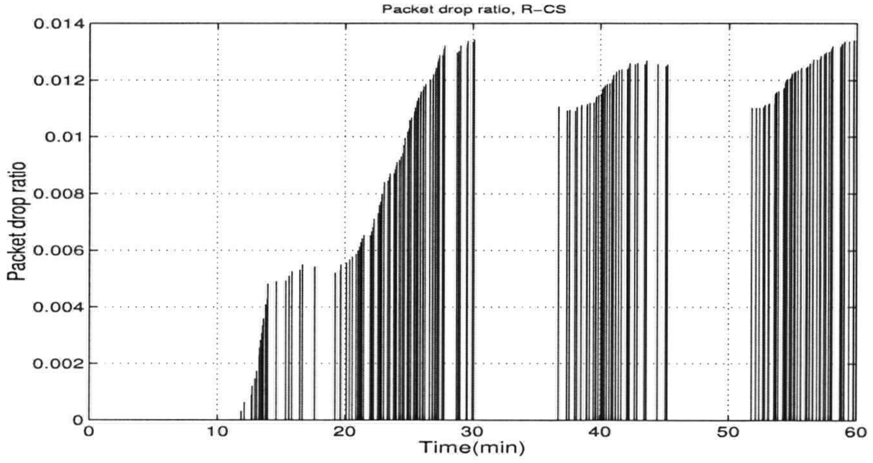

Figure 6.9: Packet drop pattern of R-CS, Load%=175%

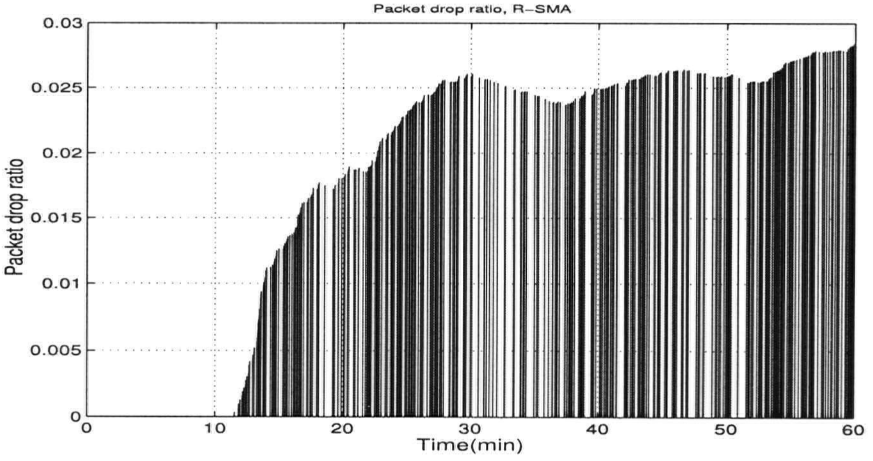

Figure 6.10: Packet drop pattern of R-SMA, Load%=175%

6.1.3 Performance of RED on a Shared-Memory Buffer with an End-to-End Congestion Manager (TCP Traffic)

The dominant transport protocol that uses an end-to-end congestion manager is TCP. The implementation of a congestion avoidance mechanism in TCP has been the main reason of proper functionality and effectiveness of the Internet, because it prevents congestion collapse phenomena. As the main purpose of RED deployment is its interaction with TCP to avoid congestion in the network, this section compares the performance of RED on a shared-memory buffer with traffic originating from an application using TCP. The simulation model is based on Figure 6.1 with four FTP traffic sources. FTP is the application running on each source, which sends and receives data through TCP and IP layers, based on the Internet's layered structure.

Similar to the traffic pattern of case B in the previous section, only two of the traffic sources attempt to send data higher than their base rates, with a base traffic rate of 1.8 kBps and added traffic of 2 kBps during 4 intervals. This traffic pattern is illustrated in table 6.3 and figure 6.11.

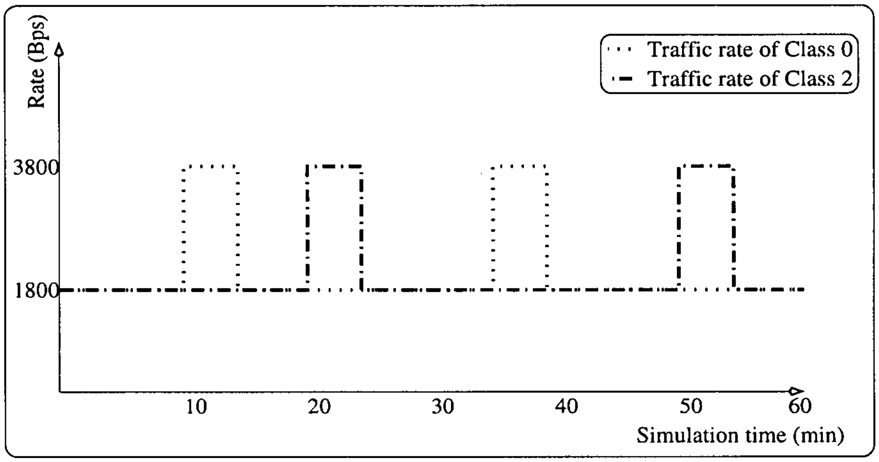

Figure 6.11: Traffic pattern of classes 0 and 2, in a simulation of one hour duration

	10 to 14	20 to 24	35 to 39	50 to 54
Class 0 (Bps)	2000	-	2000	-
Class 2 (Bps)	-	2000	-	2000

Table 6.3: Traffic pattern of TCP flows

A generic TCP model is used that emulates the performance of an actual TCP with parameter values according to table 6.4. These settings are OPNET default values. The values of the RED parameters are chosen according to table 6.5 (similar to case B in the previous section).

Initial RTO	Minimum RTO	Maximum RTO	RTT Gain
1 (sec)	0.5 (sec)	64 (sec)	0.125
Deviation Gain	RTT Deviation Coefficient	Persistence Timeout	Fast Recovery
0.25	4.0	1.0 (sec)	Enabled
Fast Retransmit	Karn's Algorithm		
Enabled	Enabled		

Table 6.4: Default values of TCP parameters used in simulation

min-th(i)	max-th(i)	Max Drop Probability	Weight Factor	Buffer size
15000 (bits)	80000 (bits)	0.1	0.002	1.25 * max-th

Table 6.5: RED parameters

The main difference between the model in this section and the one in the previous section is that when a stream suffers from a packet loss it lowers its rate, since packet loss is considered to be feedback of the network's condition, and is an indication of occurring congestion. Therefore, a combination of packet loss and throughput (traffic intended to be received versus actual traffic received) is used for performance evaluations.

Figure 6.12, 6.13 and 6.14 show samples of packet drop patterns during simulation.

Packet dropping in the R-CP schemes happens quite often and the value of the drop ratio is much higher than in the other schemes. R-SMA has more frequent drops, with higher instantaneous values in comparison to R-CS, but less drops comparing to R-CP. These figures show that in R-CS and R-SMA, packets are dropped only when the buffer is actually near or in congestion periods.

Figure 6.16, 6.15 and 6.17 show the queuing delay patterns of these sharing schemes and figure 6.18 shows the average of these figures overlaid. As the result of lower packet drops, the queuing delay of R-CS is higher than the other two, while R-CP has the lowest. At times of congestion, R-CS and R-SMA have relatively large peaks, but because of the congestion notification of RED, they return to their normal values afterward.

Figure 6.19 shows the FTP traffic received in each scenario. Since the figure shows almost similar values for the traffic received by the application layer in R-CP, R-CS, and R-SMA, and by considering the packet drop ratios, one can conclude that R-CS has the highest and R-CP has the lowest throughput.

In observing the simulation results for both TCP and non-TCP traffic, it becomes apparent that R-CS has the lowest packet dropping rates, because it uses memory more efficiently, while performing its congestion avoidance responsibility and handling bursty traffic. However, it has a bias towards non-symmetric streams, so some traffic might monopolize the buffer usage. Nevertheless, since RED drops packets proportional to the average queue size, it is less likely that a TCP-compatible class monopolize the buffer in comparison to tail-dropping CS or SMA schemes. R-SMA performs almost identically to R-CS, but with more fairness, by providing a minimum guaranteed space for each class. R-CS is the least efficient at memory usage, but is the best option for asymmetrical traffic. These algorithms can be easily extended to support weighted thresholds, and therefore, provide more fairness in addition to efficiency.

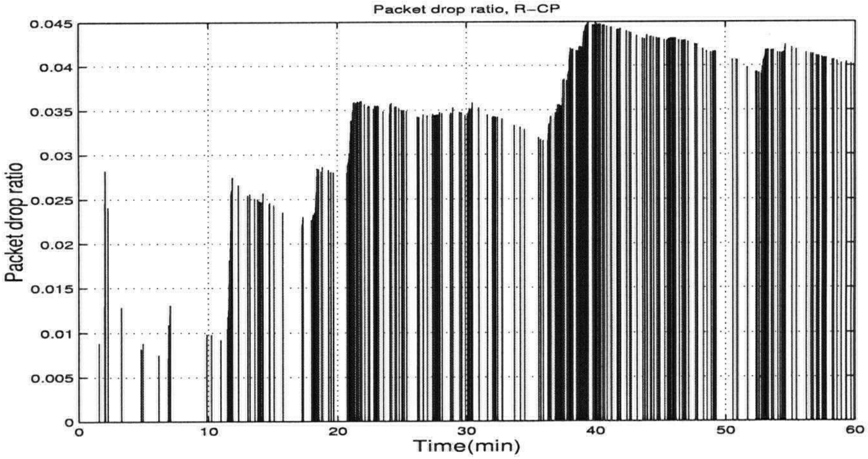

Figure 6.12: Packet drop pattern, R-CP

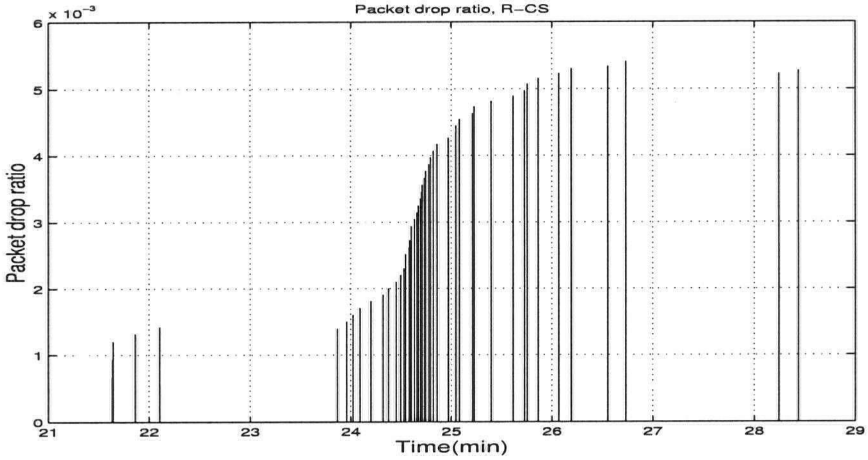

Figure 6.13: Packet drop pattern, R-CS

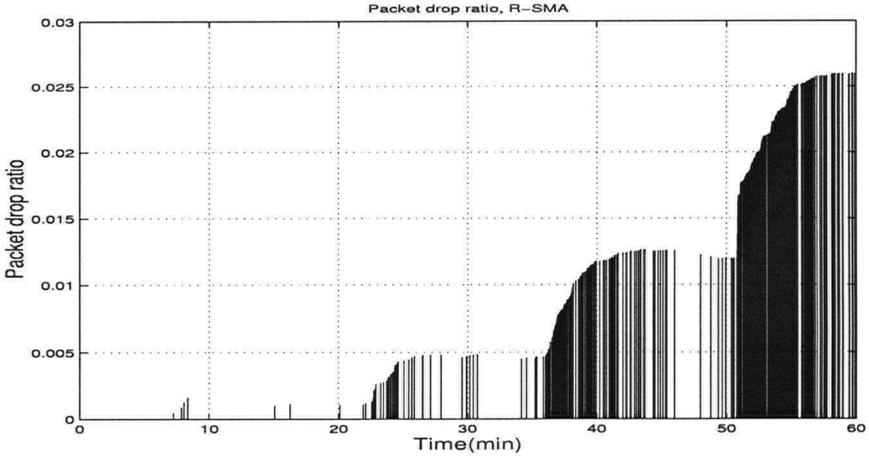

Figure 6.14: Packet drop pattern, R-SMA

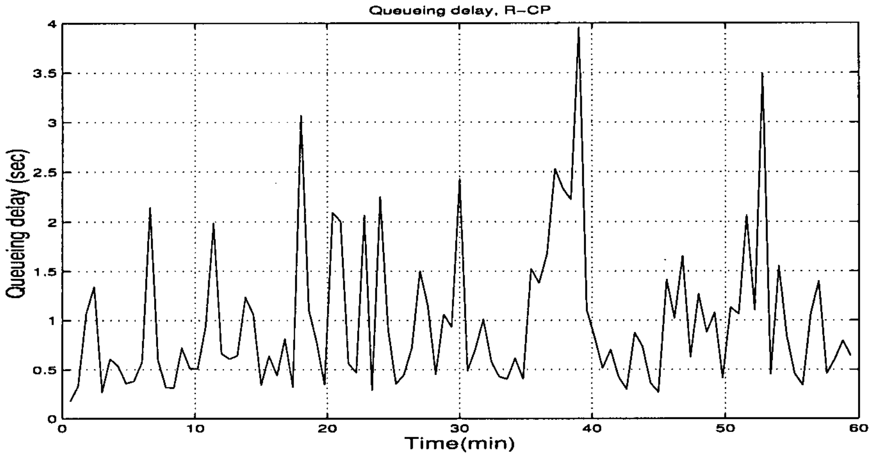

Figure 6.15: Queuing delay pattern, R-CP

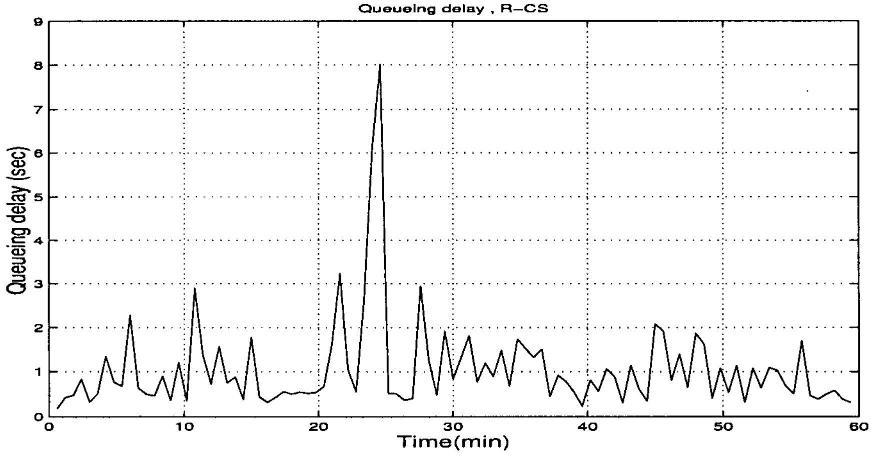

Figure 6.16: Queuing delay pattern, R-CS

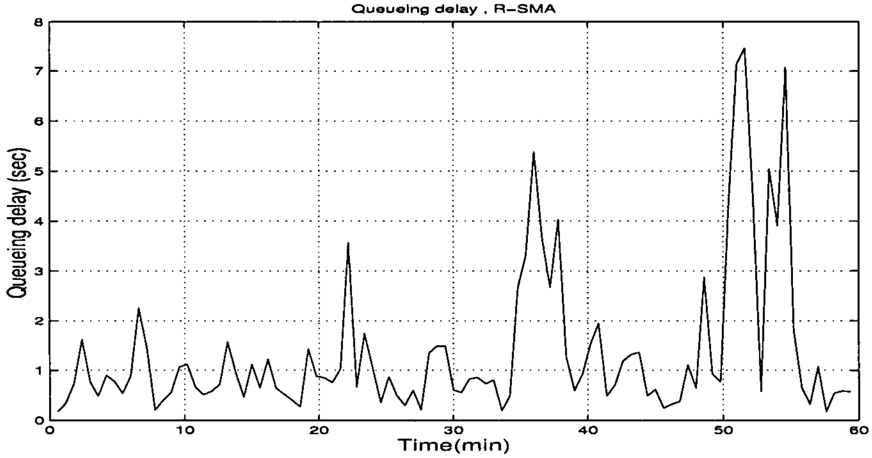

Figure 6.17: Queuing delay pattern, R-SMA

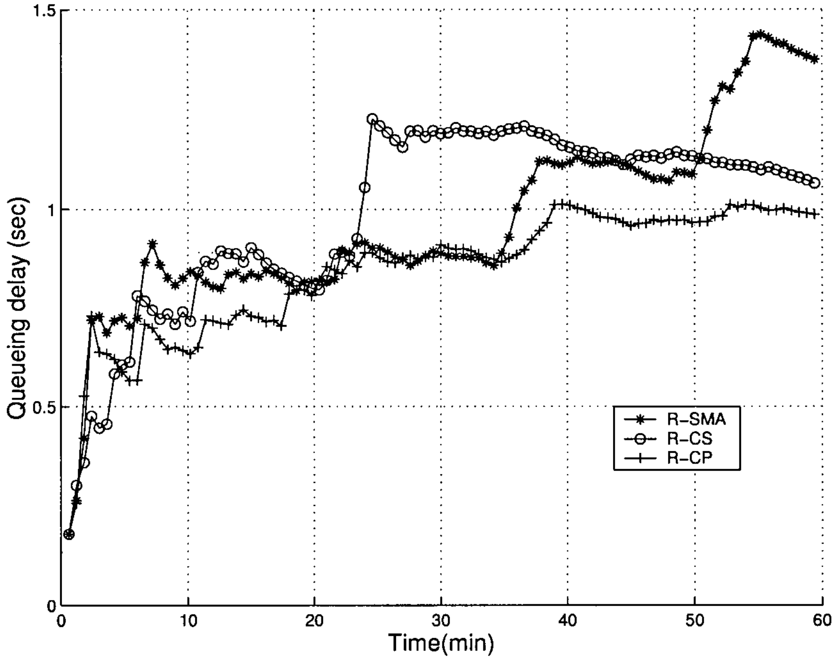

Figure 6.18: Average of queuing delay pattern

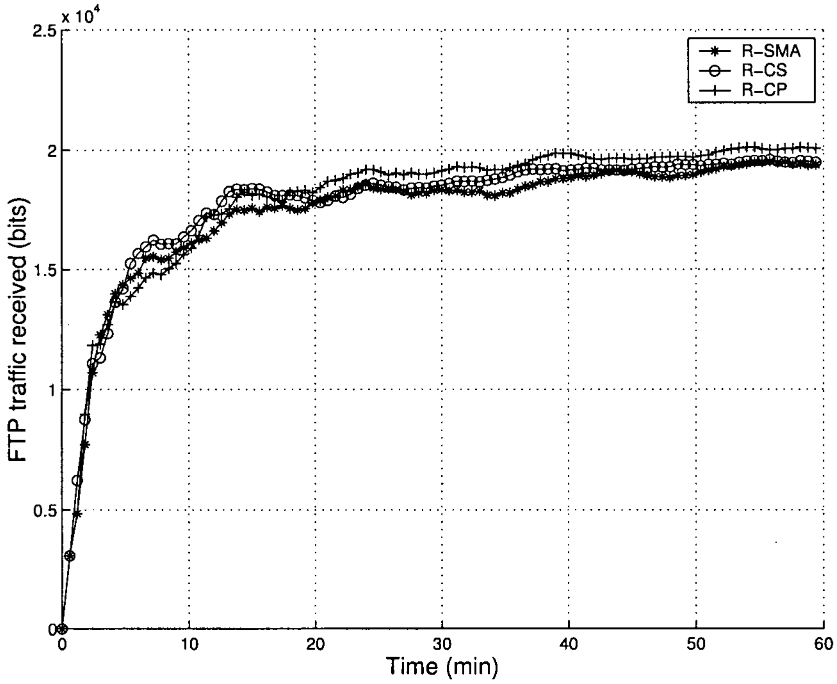

Figure 6.19: FTP traffic received

6.2 Scheduler Element

This section addresses the performance evaluation of the DDB-FFQ scheduling algorithm presented in Chapter 4. Since this algorithm is basically a modified version of FFQ, the following simulation models are aimed at evaluating the new features of this scheme, that is, the simulation results have two objectives. First,the capability of this scheme in assigning delay bounds and bandwidth allocation, independently, is shown. Second, the performance of the system when one of the streams misbehaves is assessed, which is an indicator of its link sharing ability.

6.2.1 Simulation Model

The schematic of the simulation model used is shown in Figure 6.20. The model consists of the traffic sources, the router and the destination node. The traffic sources generate four classes of flows that are multiplexed to each other in the Mux block. Each class is assigned 24% of the link bandwidth that limits the utilization to 0.97, to meet the necessary stability condition. However, they are assigned different delay bounds. Class 1 is the most delay sensitive and its delay bound is set to 1 sec and its bandwidth is 9600 bps. Class 2 is also delay sensitive and its delay bound is set to 2 sec with a 9600 bps bandwidth. Class 3 has a loose delay sensitivity of 6 seconds and Class 4 is not delay sensitive but is allocated a 9600 bps bandwidth.

An incoming packet is first classified into 4 classes and packets belonging to each of these classes are then stored in a dedicated buffer. Each stream goes through a rate-estimator module that estimates the rate of each traffic class. These collected rate statistics are fed into the scheduler. The rate estimator is an EWMA with a weight factor of $\omega = 0.02$.

The scheduler uses a DDB-FFQ scheduling algorithm. To achieve the above mentioned delay constraints, the scheduler parameters defined in Chapter 5 need to be set ap-

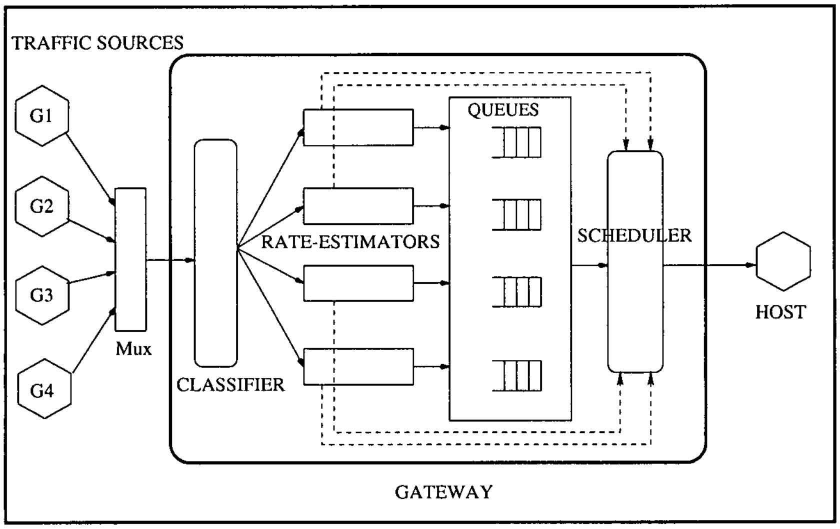

Figure 6.20: The schematic of the DDB-FFQ simulation Model

propriately, using equation (4.12) . Table 6.6 shows the values of FFQ rates and allocated rates used for the simulation.

	Class 1	Class 2	Class 3	Class 4
Allocated rates (bps)	9600	9600	9600	9600
FFQ rates (bps)	14000	11000	7000	3270

Table 6.6: The values of the FFQ and assigned rates

6.2.2 Simulation Results

To examine the decoupled delay bandwidth and link sharing ability of the DDB-FFQ system, the rate of traffic Class 2 has been set as the changing variable. It varies from 100% to 500% of its allocated rate and the observed delay and actual assigned bandwidth of each class is measured. Traffic sources 1, 3, and 4 generate packets according to their allocated rates.

Figure 6.21 illustrates the 90% delay observed by each traffic class. Since the links are ideal, the delay measured accounts only for queuing delay. The horizontal axis is the percentage of Class 2's actual rate over its assigned rate. The vertical axis is the 90% delay of the delay distribution of each traffic class over simulation duration. This figure demonstrates several facts. Misbehaving traffic has little effect on other sessions' observed delay. Class 1 delay is constant regardless of the rate of Class 2. The delay sensed by packets of traffic Class 3 varies somewhat, but its value in the worst case scenario of 500% is approximately 133% of its delay bounds. This case can be easily avoided by limiting the maximum peak rate of each traffic bandwidth to an appropriate rate. Since traffic Class 2 is misbehaving, there is no guarantee that the router will keep its promise of bounding its delay. Hence, it is punished according to its rate and its observed delay increases as its rate increases. The delay of traffic Class 4 is initially increased about 30% with the increase of the Class 2 rate, but for higher rates it can take advantage of the punishment applied to Class 2. This behavior is acceptable since the background traffic class has a loose delay requirement.

Figure 6.22 depicts the service rate received by each traffic class while Class 2 misbehaves. The vertical axis is the percentage of the actual allocated rate over the assigned rate for each class, and the horizontal axis is the percentage of the Class 2 rate over its assigned rate. It can be discerned from the figure that the allocated rate of each class varies only, at most, by 5% of its assigned rate. This fact is even applicable to the misbehaving stream. This means that the scheduler has been able to successfully limit or allocate each class's guaranteed rate. Since the bandwidth control mechanism is set to be in effect when the rate measured by the estimator is greater than 125% of what is assigned, Class 2 traffic gets higher bandwidth initially as its rate increases, and the Class 4 traffic, which has lower priority, gets lower bandwidth. Later, however, when the DDB-FFQ bandwidth control mechanism comes into effect, the trend is reversed. These figures elucidate that the only consequence for misbehaving traffic is that it will not be treated within its requested delay bound, which is a reasonable punishment.

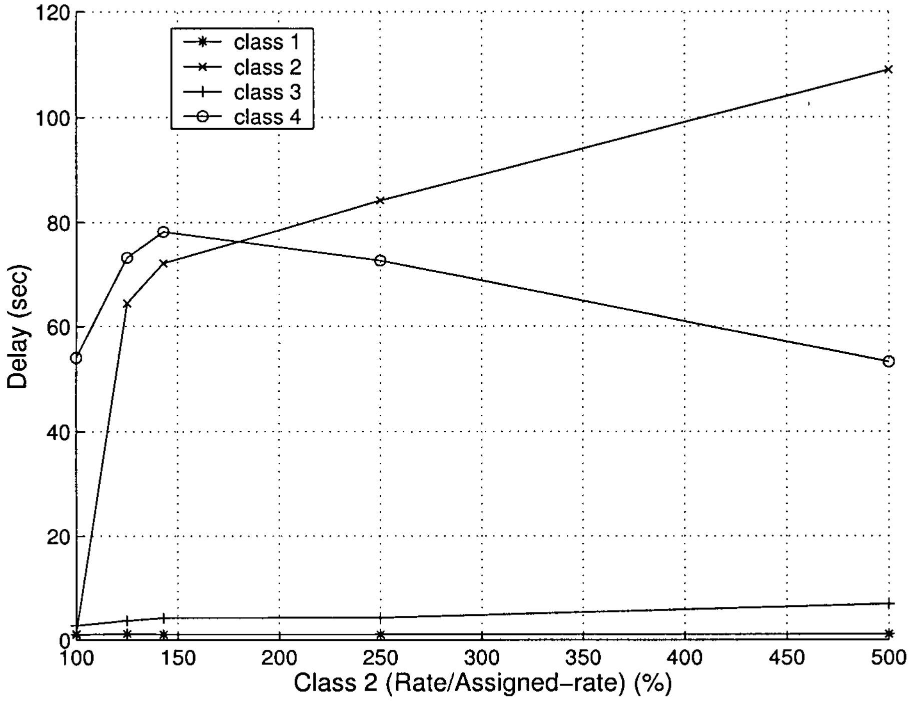

Figure 6.21: 90% delay observed by all traffic classes when class 2 is misbehaving.

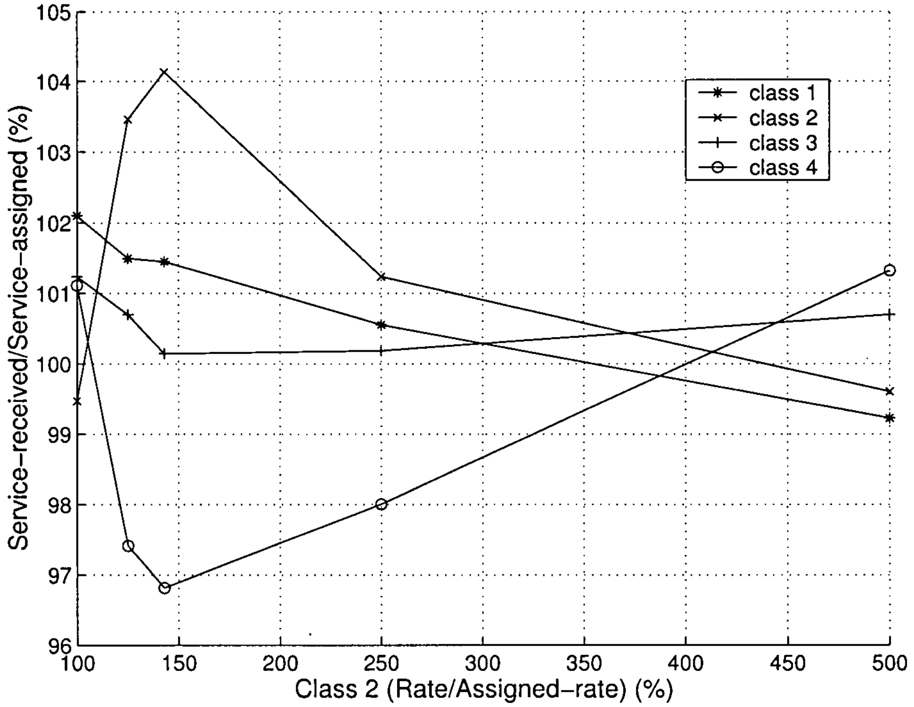

Figure 6.22: Bandwidth allocated to each class when class 2 is misbehaving.

6.3 Multiple Node Simulation Model and Results

In the previous sections, the performance of the proposed algorithms for the dropping element and the scheduler were discussed individually. In this section, the structure of a DiffServ-based UMTS backbone network is presented that has three characteristics. First, each node of the network utilizes the basic router's architecture as needed in a DiffServ router: the classifier, dropper, meter, queuing element and scheduler, with the algorithms discussed in this thesis deployed on them. The scheduler uses the DDB-FFQ algorithm by interacting with the rate-estimator module. R-CS is proposed to be used for the subgroups of a traffic aggregate, as discussed in Chapter 5. Second, the model utilizes a multiple node network consisting of an ingress node, a core node and an egress node. The ingress and egress nodes also are GGSN and SSGN, respectively. Third, the model considers all kinds of UMTS traffic classes with a more realistic traffic model. Figure 6.23 shows the OPNET network model and Figure 6.24 shows its schematic in more details.

The simulation model only considers the backbone network portion of the datapath and consists of a set of traffic sources, a simple core network, and destination nodes (Figures 6.23, 6.24). The model exploits the build-in traffic generators of OPNET to generate a traffic flow consisting of both TCP-compatible and non responsive streams. They are also chosen to include a combination of all the UMTS traffic classes; thus, it covers conversational, streaming, interactive and background classes. Five aggregates build the traffic stream going through the network.

- Network Control Traffic NC is the traffic used for signaling or network management messages that are internally used in a DiffServ domain. It occupies a very small portion of bandwidth but has the highest priority and very stringent delay sensitivity. Its required bandwidth should be considered so that its flow within the network has a very small effect on the guaranteed services of the actual DiffServ user data. Here,

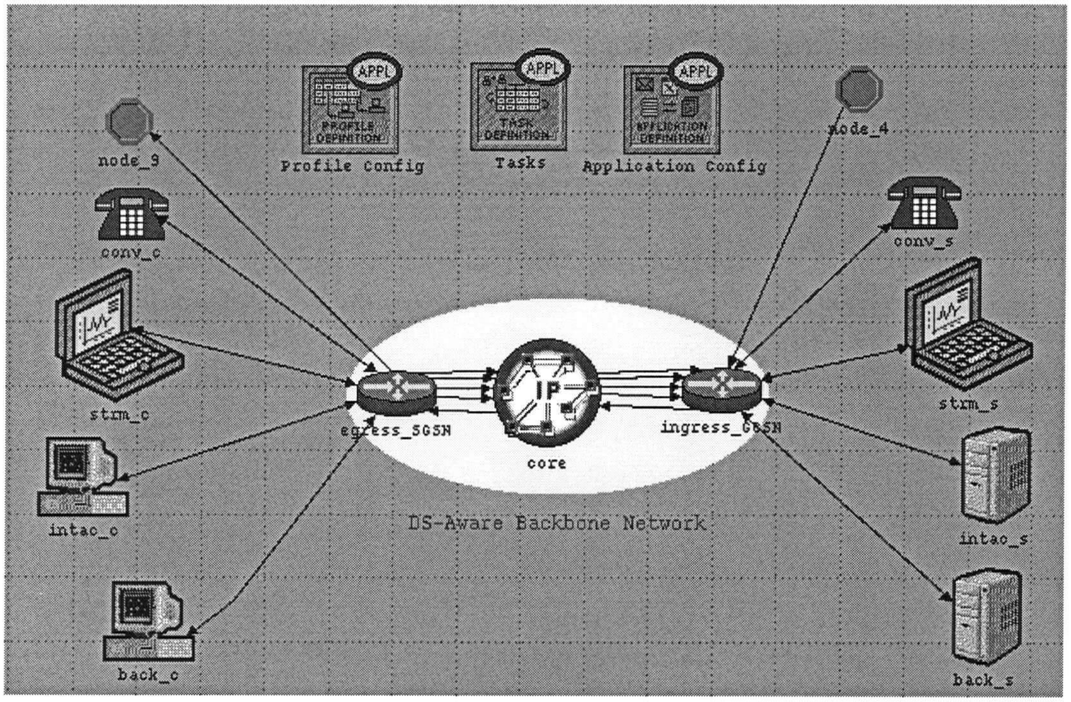

Figure 6.23: OPNET network model

it occupies 0.4% of the bandwidth and has a fixed packet size that might increase the delay bounds of the DDB-FFQ system by a small additive constant. This group is also identified by Class 0 in the simulation results.

- Conversational traffic: This traffic is an aggregate of the voice application which holds a stringent delay and jitter requirement. This aggregate is identified as Class 1 in the simulation results. Three traffic sources built this aggregate here and each source generates packets with exponentially distributed interarrival time and packet size. Although exponential distribution is not an accurate model for conversational traffic, this aggregate represents it in terms of delay sensitivity.

- Streaming traffic : This traffic aggregate that is also identified by Class 2 here, represents the aggregate of the streaming flows (video, audio streaming) in terms of delay requirements. Its rate varies from 100% of its assigned rate to 400% to evaluate the independence of the observed QoS of aggregates to each other. The traffic source out-

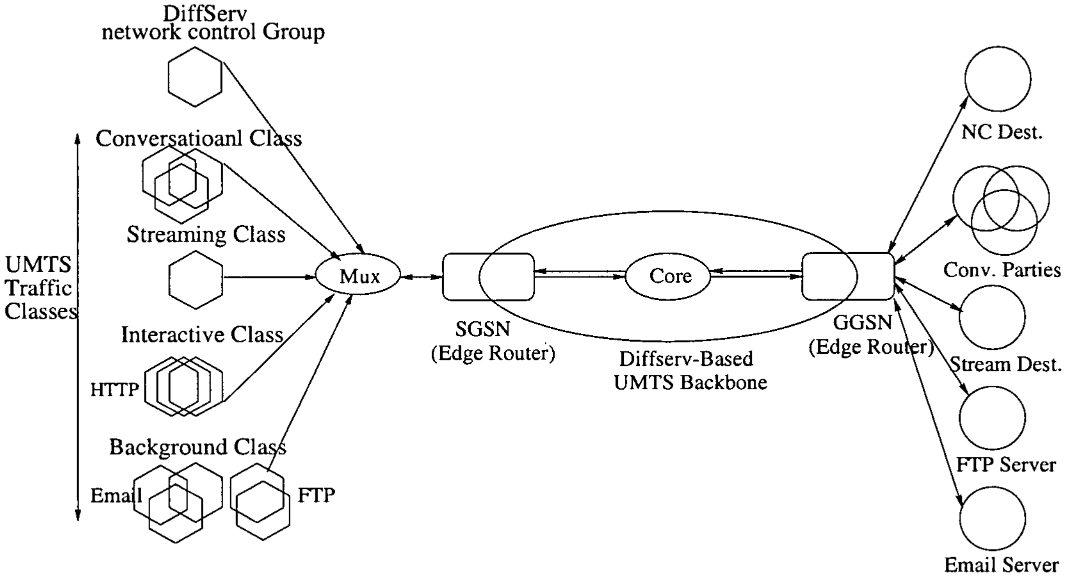

Figure 6.24: The schematic of simulation network

puts packets with exponentially distributed interarrival time and packet size.

- Interactive traffic: An aggregate of three HTTP browsing applications represents this traffic identified also as Class 3 here. The characteristics of this aggregate class are moderate delay sensitivity and contents preservation.

- Background Traffic: This aggregate consists of two instances of FTP applications and three instances of E-mail traffic, which has a low delay requisite and is identified by Class 4 in the results.

The source generators of HTTP, FTP and E-mail use the TCP/IP layered structure and represent the TCP-based traffic, while the other classes are non-responsive real-time applications.

These traffic classes have distinct delay bounds and bandwidth assignments. The bandwidth allocated to Class 0 is 0.4% of the service rate, which is a very small portion of the bandwidth. Its packet size and peak burst rate are also well defined, so that its effect on the delay bound as defined in DD-FFQ is negligible. Each of the other traffic aggregates are allocated almost the same bandwidth of about 25% of the link bandwidth. However, they

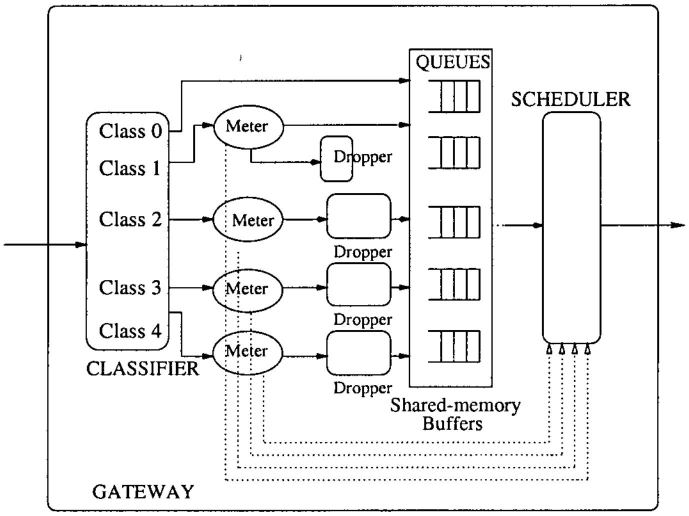

Figure 6.25: The schematic of the simulation router's configuration

have distinct delay bounds and priorities. Class 1 has the highest, and Class 4 the lowest delay requirement.

Figure 6.25 depicts the structure of every router in the model, which causes packet forwarding according to a packet's DSCP. The router has the essential components needed for proper DiffServ functionality: the classifier, the meter, the droppers, the queuing block and the scheduler. The incoming packet is first classified. If it is an edge node, its QoS is mapped to a proper PHB and then it is encapsulated, having the associated DSCP in its header. Then it is passed through a rate estimator to a dropper block. The accepted packet is then enqueued. The scheduler then forwards the packet to the next hop using the DDB-FFQ service discipline.

Dropping Element: The proposed structure for the buffer management function in our model is as follows. For each aggregate group, a dedicated buffer is allocated, that is, each of groups 1 to 5 aggregate classes has its own dedicated buffer. However, each of these buffers uses a memory sharing scheme to store the packets of the subgroups defined within each group. Group 2 uses a meter that sends the incoming packets to an absolute dropper if the average rate is greater than 120% of its assigned rate. The rest is scheduled for service under the DDB-FFQ or FFQ schemes, whether it is below its assigned bandwidth or above, respectively. Group 3 uses a tail dropping buffer and accepts packets up to its full capacity. The DDB-FFQ scheme is responsible for assigning a bandwidth to it, up to its assigned rate. Groups 4 and 5 use a RED algorithmic dropper with a RED on Complete Sharing scheme (R-CS) [48] implemented on it, since these two groups build the non-real-time aggregates.

Scheduler Element: The scheduler is a combination of the DDB-FFQ scheduling system, discussed in Chapter 4, and the absolute priority scheduling. The NC traffic aggregate has been given an absolute priority over other classes and is not scheduled under DDB-FFQ. The other classes are scheduled according to the DDB-FFQ scheme. Class 0 is treated differently for two reasons. First, it is internal network management traffic, and there is no need to have control over its bandwidth, assuming the traffic originating from the network operator is well-behaved. Therefore, using absolute priority simplifies the system. Second, it has very stringent delay requirements and needs to be serviced as soon as possible. However, this should not degrade other traffic's QoS. Nevertheless, Classes 1 to 4 are user generated and need to be scheduled according to their delay and bandwidth requirements.

A bandwidth and a delay bound are assigned to each traffic group by using the corresponding assigned rate and FFQ rate of the DDB-FFQ scheduler, respectively. Since the scheduler is a non-preemptive scheduler, delay bounds are affected by the maximum packet size of even lower priority traffic [49]. Therefore, it is necessary that the edge routers perform segmentation and re-assembly on the large packets, depending on the system delay bounds. Here, delay bounds are set by considering the 90% packet size as the maximum packet size, giving a 90% packet delay bound. Table 6.7 shows the value of internal state parameters of the DDB-FFQ scheduler, FFQ-rates and assigned rates.

Meter: The meter implementation used here is the time sliding window type. It is an exponential weighted moving average (EWMA) rate estimator [49]. This module is also

	Class 1	Class 2	Class 3	Class 4
Allocated rates (bps)	20000	20000	21200	18000
FFQ rates (bps)	42000	30000	5000	2200

Table 6.7: Values of FFQ and assigned rates

a building block of the DDB-FFQ service discipline. The mapping system used for this traffic model is summarized in table 6.8. Each of the groups can be divided into as many subgroups as needed. For example, a background traffic class can be mapped to BE, LBE, and BH PHBs. What really matters for the proper performance of the QoS system is the level of forwarding treatment each group receives and therefore, the names and codepoints can be simply implementation dependent, especially for an isolated DS network.

Group	G0	G1	G2	G3	G4
PHB	NC	EF	AF1	AF2	BE

Table 6.8: Basic mapping from UMTS to DiffServ

The core network consists of an SGSN, a core node and a GGSN. The SGSN and GGSN nodes also work as DiffServ edge routers, and they perform IP encapsulation [50] and mapping functions. The core node only applies the PHB associated with the codepoint marked in each packet header, which is a simple classification task.

The objective of the computer simulation is to evaluate the performance of the model in terms of the QoS constraints: latency, bandwidth and packet loss. Therefore, OPNET calculates and records the delay observed by each aggregate, the bandwidth assigned to each group relative to their assigned rate, and the packet loss ratio.

The independence of the service received by each traffic class is demonstrated by observing the delay and allocated bandwidth of each of them when the rate of Class 2 traffic varies from 100% to 400% of its preset rate. In a DiffServ router, the overload of misbehav-

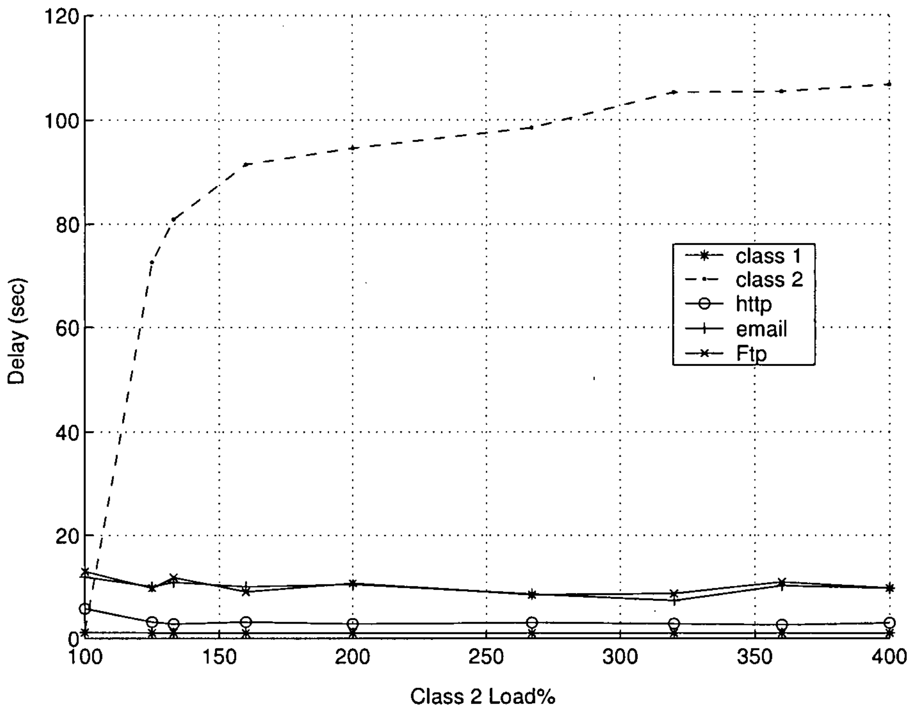

Figure 6.26: Edge-to-edge delay versus class 2 load%

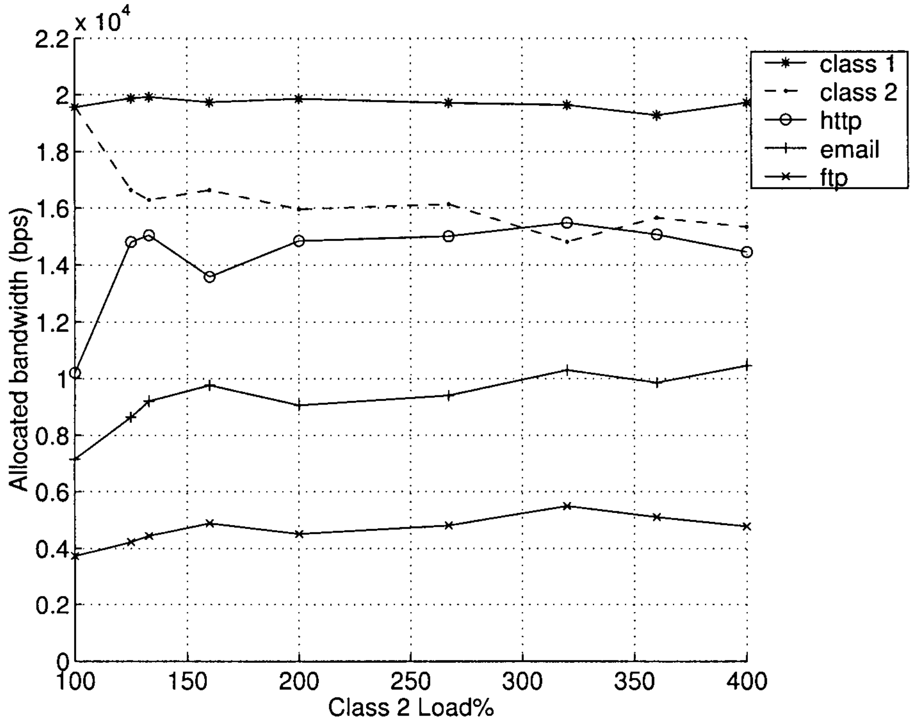

Figure 6.27: Allocated bandwidth vs class 2 load%

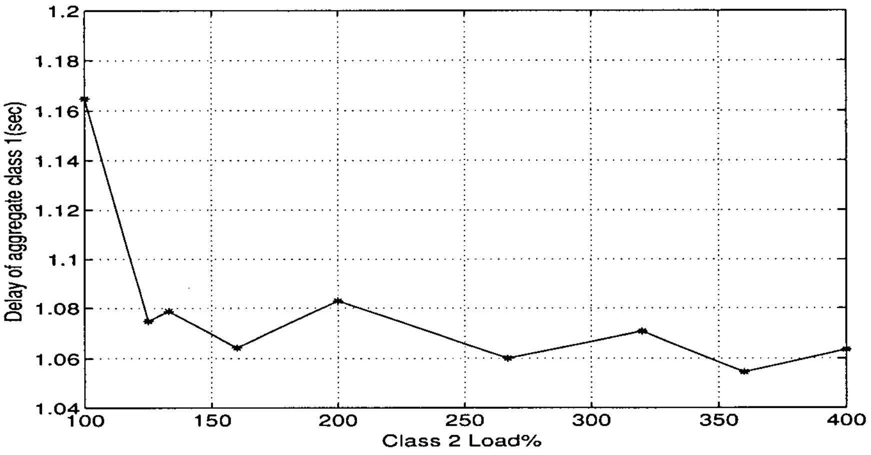

Figure 6.28: Delay of traffic class 1

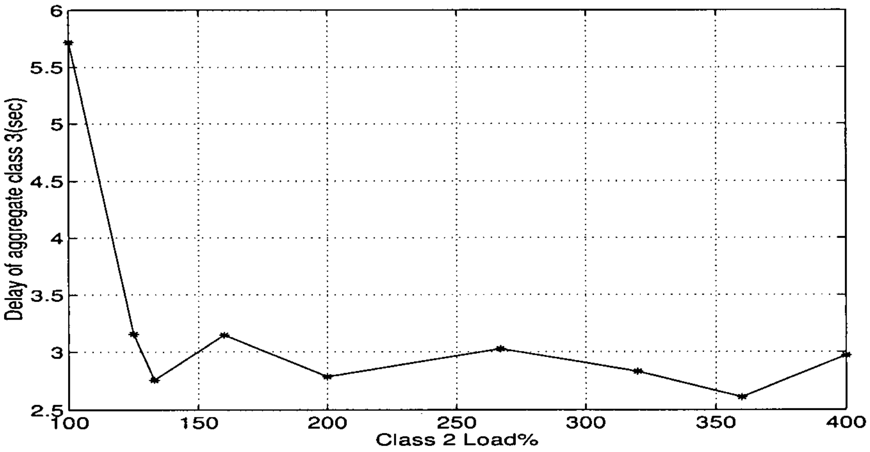

Figure 6.29: Delay of traffic class 3

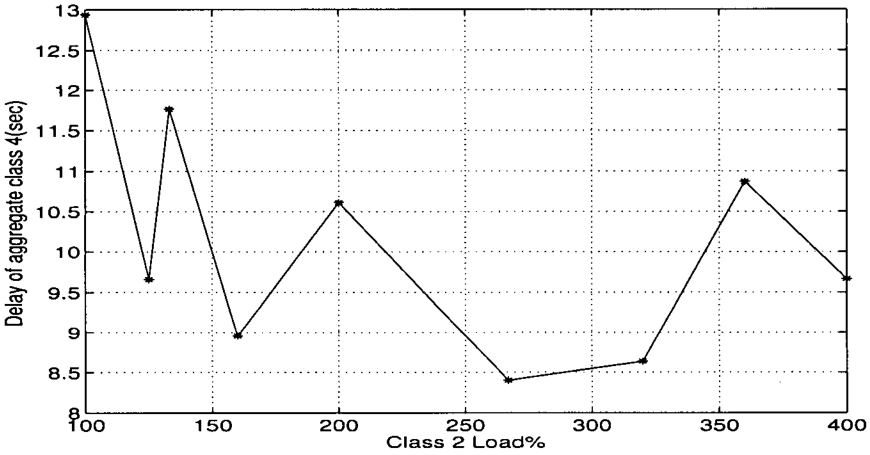

Figure 6.30: Delay of traffic class 4

ing traffic can be confronted by merely accepting the packet when the rate is below a percentage of the assigned rate, as it does for the EF PHB. However, for other kinds of traffic, for example, the streaming aggregate, it might be useful to accept packets until the corresponding queue is full, while ensuring this traffic will not degrade the service of the other streams. The size of this queue should be set to provide an acceptable maximum delay.

Figure 6.26 shows the delay observed by each application and Figure 6.27 illustrates their actual bandwidth assignments when the aggregate rate of Class 2 traffic varies from 100% to 400% of its base rate. The figures show that the PHB treatment of each aggregate is mostly independent of the behaviors of other aggregates, while they are offered a distinct range of QoS. Although the rate of traffic aggregate 2 is changing widely, the delay bound and bandwidth allocated to other traffic streams do not degrade and are changing within an acceptable range of their assigned QoS values. Class 2 itself has two bandwidth assignments, depending on if it is behaving or not. Figure 6.26 shows that the delay of each traffic aggregate does not degrade, other than the misbehaving streaming traffic.

Figure 6.27 shows an initial decrease of bandwidth assignment for streaming traffic, and correspondingly, an increase for interactive and background aggregates. This is a direct result of the method of adjustment proposed [49] for the DDB-FFQ scheduling scheme, which modifies the bandwidth according to the minimum FFQ-rate of the system. This means that the well-behaving traffic will get at least its assigned bandwidth, while misbehaving traffic is allocated a constant bandwidth, regardless of its rate, when the bandwidth control mechanism is in effect.

Figures 6.28, 6.29, and 6.30 depict the delay observed by the aggregate class 1, 3, and 4, respectively. These figures show that, initially, the delay drops until the DDB-FFQ link-sharing mechanism comes into complete effect, and then it stays almost constant.

Figure 6.31 shows the packet drop ratio of aggregate Class 2, which increases when its rate increases. The conversational traffic observes no packet dropping and aggregate

Classes 3 and 4 observe very small packet drop rates due to their RED mechanism.

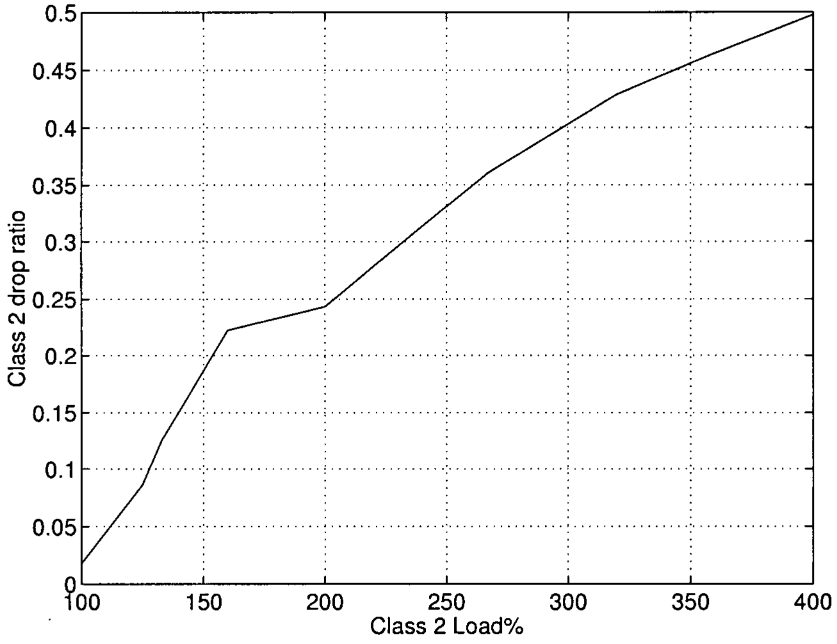

Figure 6.31: Drop ratio of aggregate class 2

Chapter 7

Conclusions

7.1 Summary

This thesis presented an insight into QoS support in the backbone network of the UMTS or GPRS system using the DiffServ model. It covered the structural model of a DiffServ router appropriate for deployment in a UMTS backbone network, in addition to providing a basic methodology for QoS mapping between UMTS and DiffServ technologies. In particular, three notable components of the router have been studied: the scheduler, the dropper, and the meter.

A novel scheduling algorithm was proposed that has the ability to provide decoupled delay bounds and bandwidth allocation. This property is an essential factor for allowing a DiffServ network operator to define and set diverse PHBs. A set of novel REDs on shared-memory schemes has also been proposed and compared, showing that their deployment will improve the performance of the system in response to congestion, as related to lower packet loss. In addition, the up-to-date QoS characteristics of both UMTS and DiffServ have been addressed and a basic mapping between these two systems has been provided. The key contributions of this thesis are as follows:

- *DDB-FFQ scheduling system.* DDB-FFQ is an efficient and simple service discipline that supports decoupled delay bound and bandwidth allocation to each aggregate class. It also obeys link-sharing principles, by providing a predictable amount of bandwidth to each traffic class during periods of congestion. This link sharing capability suits the DDB-FFQ system to a variety of other scenarios.

- *Proposal and comparison of RED on shared-memory schemes: R-CS, R-CP, R-SMA.* This thesis studies the deployment of RED on shared buffers for the first time, by deriving three RED schemes from the existent sharing policies. A brief analytical expression of RED, R-CS and R-CP is also provided.

- *QoS mapping between the two QoS systems: UMTS and DiffServ.* This thesis examines the QoS related aspects of DiffServ and UMTS, and proposes a basic QoS mapping between them. This method can be extended to include an arbitrary number of subgroup PHBs.

- *Structural study of a DiffServ-based UMTS backbone router.*

Two separate simulation models have been built to evaluate the performance of the DDB-FFQ scheduling system and the dropping algorithms. The performance of RED on shared memory buffer algorithms has been compared in terms of packet loss and queuing delay. Deployment of R-CS was proposed within the subgroups of each TCP-compatible PHB group.

Finally, a multiple node network simulation model was presented that has the following characteristics. First, it uses a traffic model consisting of both TCP-compatible and non-responsive applications, covering all UMTS traffic classes. Second, it also provides a basic, but sufficient configuration for a DiffServ router applicable in the UMTS backbone. Finally, it illustrates the just performance of the router in a multiple node model.

Although there have been some experimental deployments of GPRS and DiffServ, the actual deployments of UMTS, GPRS and DiffServ are still in the draft stage. The next section provides some potential directions for future work.

7.2 Future Work

The prospect of implementing DiffServ on the UMTS backbone network is a massive project that will require resolving a wide range of issues. Both DiffServ and UMTS technologies are evolving rapidly, and accordingly, demand ongoing research and adjustments. They are in the early stages of research and have not actually been implemented yet.

This thesis itself illuminates a number of possible future pathways. Only three DiffServ components have been taken into detailed consideration, while other elements have not been touched. The following list presents suggestions for future work within the scope of the topic presented here:

- The method DDB-FFQ applies to control the allocated bandwidth of each class is only one of numerous feasible methods. There might be a better way of achieving this objective while preserving the concept of using two sets of rates, for delay and bandwidth, as proposed for DDB-FFQ.

- There are other methods of rate estimation presented in the literature. Further studies may lead to other schemes that improve the performance of the proposed scheduling system.

- An important method of traffic management is traffic shaping, which was not touched on in this thesis. Applying a traffic shaping block to the system might provide system improvement.

- The proposed RED schemes can be extended to consider push out and/or utilize weighted thresholds parameters for RED.

Glossary

3G : 3^{rd} Generation

AF : Assured Forwarding

BE : Best Effort

BSC : Base Station Controller

CN : Core Network

DDB-FFQ : Decoupled Delay-Bandwidth FFQ

DiffServ, DS : Differentiated Services

DSCP : DiffServ CodePoint

ECN : Explicit Congestion Notification

EF : Expedited Forwarding

ETSI : European Telecommunications Standards Institute

EWMA : Exponential Weighted Moving Average

FDD : Frequency Division Duplex

FFQ : Frame-based Fair Queueing

FTP : File Transfer Protocol

GGSN : Gateway GPRS Support Node

GPRS : General Packet Radio Service

GPS : Generalized Processor Sharing

GSM : Global System for Mobile Communications

IETF : Internet Engineering Task Force

IP : Internet Protocol

Iu : UTRAN interface between the radio network controller (RNC) and CN

MT : Mobile Termination

OPNET : OPtimum NETwork performance modeler

PDB : Per Domain Behavior

PDP : Packet Data Protocol

PHB : Per Hop Behavior

PLMN: Public Land Mobile Network

PSTN : Public Switched Telephone Network

QoS: Quality of Service

R-CP : RED on Complete Partitioning

R-CS : RED on Complete Sharing

R-SMA : RED on Sharing with Minimum Allocation

RED : Random Early Detection

SAP : Service Access Point

SDU : Service Data Unit

SSGN : Serving GPRS Support Node

TCP : Transmission Control Protocol

TDD : Time Division Duplex

TE : Terminal Equipment

ToS: Type of Service

UTRAN : UMTS Terrestrial Radio Access

UMTS : Universal Mobile Telecommunications System

Bibliography

[1] G. Brasche and B. Walke, "Concepts, services and protocols of the new GSM phase 2+ general packet radio service," *IEEE Communications Magazine*, vol. 35, pp. 94–104, August 1997.

[2] ETSI TS 123 107, "UMTS QoS concept and architecture," December 2000.

[3] J. Cai and D. J. Goodman, "General packet radio service in GSM," *IEEE Communications Magazine*, vol. 35, pp. 122–131, October 1997.

[4] C. Bettstetter, H. Vogel, and J. Eberspacher, "GSM phase 2+ general packet radio service: architecture, protocols, and air interface," *IEEE Communications Surveys*, October 1999.

[5] "GPRS white paper," *http://www.Cisco.com/warp/public/cc/so/neso/gprs/gprs_wp.htm*, 2000.

[6] Y. Lina, H. C.H. Rao, and I. Clamtac, "General packet radio service: architecture, interface and development," *Wireless Communications and Mobile Computing*, vol. 1, pp. 77–92, John Wiley & Sons Journal, January-March 2001.

[7] ETSI GSM 03.60 V6.0.0, "General packet radio service; service description; stage 2," March 1998.

[8] G. Priggouris, S. Hadjiefthymiades, and L. Merakos, "Supporting IP QoS in the general packet radio service," *IEEE Network*, vol. 14, pp. 8–17, October 2000.

[9] S. Blake, D. Black, M. Carlson, E. Davies, W. Weiss, and Z. Wang, "An architecture for differentiated services," *RFC 2475*, December 1998.

[10] D. Black, S. Brim, B. Carpenter, and F. Le Faucheur, "Per hop behavior identification codes," *IETF Internet Draft (revision of RFC 2836)*, January 2001.

[11] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the differentiated services field in the IPv4 and IPv6 header," *RFC 2474*, December 1998.

[12] ISI University of Southern California, "Internet protocol," *RFC 791*, September 1981.

[13] Y. Bernet, A. Smith, S. Blake, and D. Grossman, "A conceptual model for diffserv routers," *IETF Internet Draft*, May 2000.

[14] Y. Bernet, S. Blake, D. Grossman, and A. Smith, "An informal management model for diffserv routers," *IETF Internet Draft*, February 2001.

[15] V. Jacobson, "Congestion avoidance and control," *Proceedings of ACM conference of the Special Interest Group on Data Communications*, pp. 314–329, Standford, September 1988.

[16] S. Floyd and V. Jacobson, "Random early detection gateway for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, August 1993.

[17] R. Jain and K.K. Ramakrishnan, "Congestion avoidance in computer networks with a connectionless network layer: concepts, goals and methodology," *Proceedings of the IEEE Computer Networking Symposium*, pp. 303–313, Maryland, MA, April 1988.

[18] H. Balakrishnan and S. Seshan, "The congestion manager," *RFC 3124*, June 2001.

[19] T. Boland, M. May, and J. C. Bolot, "Analytic evaluation of RED performance," *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 415–1424, Tel Aviv, Israel, March 2000.

[20] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification to IP," *RFC 2481*, January 1999.

[21] B. Braden *et all*, "Recommendations on queue management and congestion avoidance in the internet," *RFC 2309*, April 1998.

[22] R. Morris, E. Kohler, J. Jannotti, and m. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, pp. 263–297, August 2000.

[23] I. Cidonn and R. Guerin, "Optimal buffer sharing," *IEEE Transactions on Selected Area in Communications*, vol. 13, pp. 1229–1240, September 1995.

[24] M. Arpaci and J. A. Copeland, "Buffer management for shared-memory ATM switches," *IEEE Communications Surveys*, First Quarter 2000.

[25] A.K. Choudhury and E.L. Hahne, "Dynamic queue length thresholds for shared-memory packet switches," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 130–140, April 1998.

[26] R. W. Wolff, *Stochastic modeling and the theory of queues*, Prentice Hall, 1989.

[27] D. Bertsekas and R. Gallager, *Data networks*, Prentice Hall, 1992.

[28] A. Papoulis, *Probability, random variables and stochastic processes*, McGrow-Hill, 3rd edition, 1991.

[29] H. Zhang, "Service disciplines for guaranteed performance service in packet switching networks," *Proceedings of the IEEE*, vol. 83, pp. 1374–1396, October 1995.

[30] D. Stiliadis and A. Verma, "Rate-proportional servers: a design methodology for fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 164–174, April 1998.

[31] S.S. Lam and G.G. Xie, "Group priority scheduling," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 205–218, April 1997.

[32] A. K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 344–357, June 1993.

[33] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 365–386, August 1995.

[34] S. Floyd, "Notes on CBQ and guaranteed services," *www.aciri.org/floyd/cbq.html*, July 1995.

[35] I. Stoica, H. Zhang, and T.S.E. NG, "A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 185–199, April 2000.

[36] H. Sariowan, R.L. Cruz, and G.C. Polyzos, "SCED: a generalized scheduling policy for guaranteeing quality of service," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 669–684, October 1999.

[37] H. Sariowan, R. Cruz, and G.C. Polyzos, "Scheduling for quality of service guarantees via service curves," *Proceedings of IEEE Fourth International Conference on Computer Communications and Networks*, pp. 512–520, Seattle, WA, September 1995.

[38] D. Stiliadis and A. Verma, "Efficient fair queueing algorithms for packet switched networks," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 175–185, April 1998.

[39] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," *RFC 2598*, June 1999.

[40] A. Charny, F. Baker, J. Bennett, K. Beson, J. Le Boudec, A. Chiu, W. Courtney, B. Davie, and S. Davari, "EF PHB redefined," *IETF Internet Draft*, November 2000.

[41] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawsk, "Assured forwarding PHB group," *RFC 2597*, June 1999.

[42] R. Bless and K. Wehrle, "A lower than best-effort PHB," *IETF Internet Draft*, September 1999.

[43] M. Loukola, J. Ruutu, and K. Kilkki, "Dynamic RT/NRT PHB group," *IETF Internet Draft*, November 1998.

[44] B. Carpenter and K. Nichols, "A bulk handling PDB for differentiated services," *IETF Internet Draft*, January 2001.

[45] P. Hurley, "The alternative best-effort service," *IETF Internet Draft*, June 2000.

[46] N. Seddigh and J. heinanen, "An assured rate per domain behavior for differentiated services," *IETF Internet Draft*, December 2000.

[47] S. Floyd, "Discussion of setting parameters," *www-nrg.ee.lbl.gov/floyd/red.html.*

[48] F. Agharebparast and V.C.M. Leung, "Improving the performance of RED deployment on a class based queue with shared buffer," *proc. IEEE Global Telecommunications Conference*, San Antonio, Texas, November 2001.

[49] F. Agharebparast and V.C.M. Leung, "Efficient fair queueing with decoupled delay-bandwidth guarantees," *proc. IEEE Global Telecommunications Conference*, San Antonio, Texas, November 2001.

[50] D. Black, "Differentiated services and tunnels," *RFC 2983*, October 2000.

[51] L. L. Peterson and B. S. Davie, *Computer networks: a system approach*, Morgan Kauffmann Publishers, 2nd edition, 2000.

[52] A. S. Tanenbaum, *Computer networks*, Prentice Hall, 2000.