INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

QoS Control for IP Networks: Buffer Management and Backward Explicit Congestion Notification(BECN)

by

Rupinder Singh Makkar

**B.Eng.(Electronics and Communication)
Punjab University, Chandigarh, India**

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

Master of Engineering

**Ottawa-Carleton Institute for Electrical and Computer Engineering
Faculty of Engineering**

**Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, K1S 5B6, Canada**

December 18, 2000

©Rupinder Singh Makkar

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

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

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

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

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

0-612-61021-7

Canada

The undersigned recommend the Faculty of Graduate
Studies and Research the acceptance of the thesis

**QoS Control for IP Networks: Buffer Management
and Backward Explicit Congestion Notification (BECN)**

Submitted by Rupinder Singh Makkar, B.Eng.
in partial fulfillment of the requirements for the degree of
Master of Engineering

Professor Ioannis Lambadaris

Thesis Supervisor

A handwritten signature in black ink, appearing to be 'Ioannis', written over a horizontal line.

Professor J.W. Chinneck

Chair, Department of Systems and Computer Engineering

A handwritten signature in black ink, appearing to be 'J.W. Chinneck', written over a horizontal line.

Abstract

The IETF Differentiated Services Working Group has recently standardized the Assured Forwarding (AF) Per Hop Behavior (PHB). RFC 2597 recommends that an Active Queue Management (AQM) technique be used to realize the multiple levels of drop precedence required in the AF PHB. The most widely used AQM scheme is RED (Random Early Detection). There are several ways to extend RED to a Multi-level RED (MRED) algorithm suitable for the AF PHB. This thesis compares two possible MRED implementations and their ability to protect traffic with lower drop precedence: WRED (Weighted RED) and RIO (RED with in/out).

Based on an empirical study, this thesis makes the following key contributions: Firstly, the results show that for ON-OFF traffic, RIO is better than WRED in protecting packets marked for treatment with lower drop precedence. Secondly, for short-lived flows, RIO achieves higher transactional rates than WRED. Thirdly, for bulk transfer, RIO and WRED achieve comparable long-term throughput.

The second part of this thesis evaluates a congestion control scheme involving the use of routers with existing end-to-end congestion control i.e. BECN (Backward Explicit Congestion Notification). Behavior of BECN is compared with ECN and Non-ECN (TCP RENO) with both bulk and transactional transfers. Experiments in this thesis show a definite improvement of BECN over TCP RENO in both cases of data transfers. However, in comparison with ECN, BECN is found to give mixed results.

Acknowledgments

I am sincerely grateful to my thesis supervisor, Professor Ioannis Lambadaris for his guidance, encouragement, motivation and continuous support in my research. His vast knowledge and experience in the area of Networking proved very useful during the research.

My utmost thanks goes to Jamal Hadi Salim, Nabil Seddigh and Biswajit Nandy, OPEN IP Group, Nortel Networks. This research was made possible by their help, support and rightful guidance.

My colleagues at Broadband Networks Lab have been very helpful and supportive for which I am thankful to them. At the same time, I would like to pay my gratitude to all those who have helped me in this research. I am also thankful to my parents and family members for their support and understanding during the course of my studies.

Finally I would like to thank Communications and Information Technology, Ontario and Late Professor Roger Kay's family for their financial contributions towards this research.

Contents

1	Introduction	1
1.1	Background and Motivation	1
1.2	Objective of the Thesis	4
1.3	Thesis Outline	5
1.4	Thesis Contributions	6
2	Differentiated Services	8
2.1	Introduction	8
2.2	Components of a Differentiated Services IP Network	9
2.2.1	Network Policy/Resource Management	10
2.2.2	Edge device functionality	10
2.2.3	Core device functionality	11
2.3	Type of Services On Diffserv Network	13
2.4	MRED- Multi-Level RED	14
2.4.1	Taxonomy	15
2.4.2	WRED - Weighted RED	16
2.4.3	RIO-C (RED with In/Out and Coupled virtual Queues)	16
2.4.4	RIO-DC (RED with In/Out and De-Coupled Queues)	18
3	Experimental Setup and Test Descriptions	19

3.1	Experimental Configuration	20
3.2	Experimental Descriptions and Results	21
3.2.1	Experiment 1: ON-OFF bursty traffic	22
3.2.2	Experiment 2: Transactional transfers	27
3.2.3	Experiment 3: Bulk transfers	30
3.2.4	Experiment 4: Varying RED parameters for WRED	34
3.3	Conclusions	34
3.4	Future Work	37
4	Congestion Control in TCP	38
4.1	Introduction	38
4.2	Queue management in routers	39
4.2.1	Active Queue Management	40
4.2.2	Random Early Detection	40
4.3	Explicit Congestion Notification	41
4.3.1	Implementations issues for ECN	42
4.3.2	Functional Description of ECN	42
4.4	Backward Explicit Congestion Notification	44
4.4.1	Drawbacks of ECN	45
4.4.2	Implementation issue for BECN	46
4.4.3	BECN Implementation- Router	47
4.4.4	BECN Implementation-Source End Hosts	48
5	Validation, Terminology used and Parameter selection	50
5.1	Validation of BECN	50
5.2	Terminology Used	51
5.3	RED parameter selection	53

6 Experiments, Results and Analysis	57
6.1 Experiment 1: Heterogeneous Network with different congestion levels . .	57
6.1.1 Results and Analysis	59
6.2 Experiment 2: Homogeneous network: Varying Number of flows	63
6.2.1 Results and Analysis	64
6.3 Experiment 3: Varying RTT with fixed number of flows	66
6.4 Experiment 4: Short Web Transfers	67
6.4.1 Result and Analysis	68
6.5 Conclusions	69
6.6 Future Work	70

List of Figures

2.1	Sample Network Configuration for DS	12
2.2	Variants of RED	15
2.3	Parameter setting options for MRED	17
3.1	Experimental Setup	20
3.2	Staggered RED parameter setting	23
3.3	Queue variations for ON-OFF traffic: RIO-C	23
3.4	Queue variations for ON-OFF traffic: WRED	24
3.5	Partially Overlapped RED parameter settings	25
3.6	Overlapped RED parameter settings	26
4.1	Figurative description of RED	41
4.2	Pictorial representation of ECN	44
4.3	Pictorial representation of ECN and BECN	49
5.1	Sequence versus time graph for BECN	51
5.2	Congestion window corresponding to Figure 5.1 BECN flow	52
5.3	Queue Variation under Low Congestion for Heterogeneous Network . . .	54
5.4	Queue Variation under Medium Congestion for Heterogeneous Network .	54
5.5	Queue Variation under High Congestion for Heterogeneous Network . . .	55
5.6	Queue Variation for 5 flows in a Homogeneous Network	55

5.7	Queue Variation for 25 flows in a Homogeneous Network	56
6.1	Network Topology for experiment 1	58
6.2	Goodput for Non-ECN,ECN,BECN under Low,medium and high congestion	59
6.3	Packet drops for Non-ECN, ECN, BECN under Low,medium and high congestion	60
6.4	Congestion window for BECN for low congestion (microscopic view) . . .	61
6.5	Congestion window for ECN for low congestion (microscopic view) . . .	61
6.6	Congestion window for ECN for high congestion (microscopic view) . . .	62
6.7	Network Topology for experiment 2	63
6.8	Average Goodput for Non-ECN, ECN, BECN with varying number of flows	64
6.9	Packet drops for Non-ECN, ECN, BECN with varying number of flows .	65
6.10	Individual goodput of each flow for Non-ECN, ECN, BECN	66
6.11	Packet drops for Non-ECN, ECN, BECN with variable RTT	67
6.12	Transfer time distribution for 100 short web transfers	68

List of Tables

3.1	Tests Performed	22
3.2	ON-OFF bursty traffic- Staggered RED	24
3.3	ON-OFF bursty traffic- Partially overlapped RED	26
3.4	ON-OFF bursty traffic: Overlapped RED	27
3.5	Transactional transfers: Staggered RED parameters	28
3.6	Transactional transfers: Partially Overlapped RED parameters	29
3.7	Transactional transfers: Overlapped RED	30
3.8	Bulk Transfers: Staggered RED parameters	32
3.9	Bulk Transfer: Partially Overlapped RED parameter	32
3.10	Bulk transfers: Overlapped RED parameters	33
3.11	WRED with Grossly Staggered Parameters	35
6.1	RED parameter settings	58
6.2	RED settings: Experiment 2, 3, 4	63

Chapter 1

Introduction

1.1 Background and Motivation

Quality of Service on the Internet has become an important issue due to upsurge in the applications using Internet. On the Internet and in other networks, QoS (Quality of Service) is the idea that transmission rates, error rates, and other characteristics can be measured, improved, and to some extent, guaranteed in advance. It relates with the issues like latency, bandwidth utilization, packet drops etc. QoS is of particular concern for the transmission of voice, high-bandwidth video and multimedia information. Transmitting this kind of content dependably is difficult in public networks using ordinary best effort protocols. This thesis focuses on two of related issues: Buffer Management for Differentiated Services and Congestion Control.

Differentiated Service (Diffserv) mechanism allow service providers to allocate different levels of service to different users of the Internet. It tries to offer differing level of treatment to user traffic based on their requirements. Diffserv is based on premise that complicated functionality should be moved towards edge of the network with very simple functionality at the core. A local host or a first hop router can act as an Edge

device. Edge devices in Diffserv architecture are responsible for marking and aggregating the packets based on service class and core devices gives them differentiated aggregated treatment based on packet marking done by edge devices.

Active Queue Management scheme like RED (Random Early Detection) is likely to be used at the core for differential treatment. RED works based on an average queue which is calculated using exponential weighted average of instantaneous queue. Two set of thresholds min_{th} and max_{th} are maintained for drop decision. Multi-level RED (MRED) is a generic term used to describe any scheme where drop probabilities of packets of different drop precedence (in reference to diffserv) is calculated differently. RIO-C (RED with IN/OUT and coupled queues) and WRED (Weighted RED) are two variants of MRED. A more detailed description of MRED can be found in next chapter.

In [6] May et al. perform some analytical modeling of Differentiated Service Architecture schemes. Firstly, using throughput as the performance indicator, they find that it does not matter whether the acceptance probability of tagged [marked] packets depends on the total number of packets or on the number of tagged packets in the queue. This thesis investigates this conclusion further using other performance indicators such as packet loss with a wider variety of traffic etc. Secondly, based on analytic evaluation of the loss probability, they conclude: choice of different parameter values can have a clear impact on performance. This conclusion is investigated in detail as a part of this thesis.

Congestion Control is another issue which directly affects the Quality of Service that can be provided on the Internet. Congestion results in packet drops, large packet delays and degradation in network utilization. Congestion can occur when data arrives on a big pipe (a fast LAN) and gets sent out over a smaller pipe (a slower WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs [24]. The traditional way that a router reacts to congestion

is by dropping the incoming packets in absence of buffer space. This is referred to as *tail drop*. This dropping of a packet in TCP/IP network is taken as a sign of congestion in the network.

TCP in response to this indication of congestion invokes its end-to-end congestion control algorithms [26][27][28] to alleviate congestion. Although end-to-end congestion control schemes have served the Internet well, with the increase in the amount of traffic and congestion on the Internet, some improvements in these end-to-end congestion control has become necessary. Such existing schemes treat the network as blackbox. There is a limit to how much control can be accomplished from the edges of the network, so need to complement these congestion control schemes. The congestion control algorithms at routers is one of the way to complement the already existing algorithms.

Queue management, is one of the router algorithm that can be used in conjunction with the end-to-end congestion control schemes. Queue management deals with managing the lengths of packet queues by dropping packets when necessary or appropriate (e.g. in case of congestion). Random Early Detection is one of the technique that has gained popularity in recent years. It's based on detecting incipient congestion by dropping packets probabilistically depending upon the values of threshold parameters. This causes the sender TCP to back off and hence alleviate congestion before it actually occurs.

Explicit Congestion Notification (ECN) [29] and Backward ECN (BECN) [30] are two of the recently proposed congestion control algorithms, which uses end-to-end as well as router support for congestion control. They mark the packet based on RED, instead of dropping it. This reduces the number of packet drops, which benefits TCP during time of congestion. More detailed of ECN and BECN is provided in chapter 4.

Ahmed et al [38] did evaluation of ECN and BECN on Linux based setup with real network traffic. Experiments were done with different congestion levels and by varying RED parameters. They showed that ECN is able to perform better than Non-ECN in

terms of throughput and transaction rate. Further they have shown BECN to be even better than ECN in terms of these parameters.

1.2 Objective of the Thesis

This thesis presents a empirical evaluation of two topics related to Quality of Service. We did empirical evaluation of two most used buffer management scheme (WRED and RIO-C) for Diffserv Assured Forwarding PHB in the first issue. The second issue evaluated a new proposed congestion control scheme in IP Networks against the existing congestion control schemes. The specific objective of this thesis are to :

- Setting up of a real testbed based on Vxworks and Linux for evaluation of WRED and RIO-C.
- Empirical evaluation of WRED and RIO-C based on following parameters:
 - Ability of these two schemes to protect lower Drop Precedence traffic.
 - Transaction rate for Web type short transfer traffic.
 - Throughput for Bulk transfer traffic.
- Validation of above results using Network Simulator (NS).
- Implementation of BECN in NS using C++, TCL and OTCL.
- Evaluation of Non-ECN, ECN, and BECN based congestion control schemes on NS under following conditions:
 - Bulk transfers under high, medium, and low congestion levels in a heterogeneous network. The parameters used are goodput and packet drops.

- Bulk transfers in a homogeneous network with fixed RTT but varying the number of flows in the network. Same parameters of goodput and packet drops are used for evaluation.
- Varying RTT with fixed number of flows for a homogeneous network.
- Small duration short transfers in a homogeneous network. Transaction rate/transfer time and packets drops are used for evaluation.

1.3 Thesis Outline

Rest of the thesis is organized as follows:

Chapter 2: Reviews fundamentals of Diffserv Architecture. Brief functional description of various components of Diffserv i.e. Edge Devices, Core devices, Policy management follows. Last portion of this chapter describes different types of Multi-Level RED (RIO-C, WRED, RIO-DC) based buffer management schemes.

Chapter 3: Describes the experimental testbed along with the set of parameters used for evaluation of WRED and RIO-C. These have been evaluated under different type of RED parameter settings and network traffic. Experimental results are presented in tabulated as well as graphical form. Based on results obtained, discussion, conclusion and related future work is also described at the end of this chapter.

Chapter 4: Fundamentals of Traditional Congestion control (DropTail), Explicit Congestion Notification (ECN) and Backward ECN (BECN) in TCP/IP is summarily described. Drawbacks of these schemes along with the implementation issues for ECN and BECN have also been presented in this chapter.

Chapter 5: Validation of BECN implementation is covered. Definition of various terminologies used in subsequent chapter is provided. RED parameter selection is

explained in detailed manner.

Chapter 6: Description of the experiments carried out under different network topologies, flows, and RTT with Non-ECN, ECN, and BECN as congestion control schemes. Results are presented in tabulated and graphical form. This is followed by discussion, conclusion, and related future work to be done.

1.4 Thesis Contributions

The following are the main contributions of this thesis:

- Detailed Empirical evaluation of WRED and RIO-C for Differentiated Services Assured Forwarding PHB with focus on following issues:
 - Ability of these two schemes to protect lower Drop Precedence traffic.
 - Transaction rate for Web type short transfer traffic.
 - Throughput for Bulk transfer traffic.
- RIO-C is found to outperform WRED for all experimental test cases. This study resulted in a paper “Empirical Evaluation of Buffer Management schemes for Diff-serv Assured Forwarding PHB” presented at ICCCN 2000, October 16-18, Las Vegas, USA.
- Implementation and validation of BECN in Network Simulator.
- Evaluation of BECN based congestion mechanisms against ECN and Non-ECN congestion mechanisms under following test cases:
 - Bulk tranfers under varying level of congestions and flows.
 - Small duration short transfers.

- Study showed definite improvements for BECN over Non-ECN under all test cases. However, mixed results of Improvement/Degradation for BECN in comparison with ECN were obtained. This work resulted in an internal report “NS implementation of BECN” for Nortel Networks for providing a reference to carry on further research/improvements for BECN.

Chapter 2

Differentiated Services

2.1 Introduction

Traditional IP networks offer users Best-Effort service. In this model, all packets compete equally for network resources. The best-effort model has been sufficient until recently when the usage and popularity of IP networks has soared. The rise in usage and popularity of the internet has been paralleled by a rise in user expectation regarding the type and quality of application that can be run on these networks. However due to limited network resources (bandwidth, buffer constraints), sporadic and periodically sustained congestion has been observed on current IP networks.

Congestion of this nature does not encourage mass adoption of IP networks as a transport mechanism for Real time and business deemed applications. Developing improved and reliable service predictability for IP networks is one of the key challenges. The underlying concept in IP Quality of Service (QoS) is the ability of network operators to offer differing levels of treatment to user traffic based on their requirements.

The Differentiated Services (Diffserv) approach proposes a scalable way to deliver IP QoS. Diffserv architecture [3] has recently become the preferred method to address QoS issues in IP networks. It operates on the premise that complicated functionality should

be moved towards the edge of network with very simple functionality at the core of the network. Edge devices in this architecture are responsible for ensuring that individual user traffic conforms to traffic profiles specified by network operator and for grouping flows in an aggregated fashion into a small number of classes. Core devices perform differentiated aggregate treatment of these classes based on the marking performed by the edge devices.

An end-to-end differentiated service is obtained by concatenation of per-DS domain services and Service Level Agreements (SLAs) [3] between adjoint DS domains along the path that the traffic crosses in going from source to destination. DS domain, a Diffserv Architecture term, refers to a contiguous set of nodes which operate with a common set of service provisioning policies and Per Hop Behavior (PHB) definitions. Per hop behavior refers to the external treatment applied to a packet at a node. Per domain services are realized by traffic conditioning [3] at the edge and simple differentiated forwarding mechanisms at the core of the network.

2.2 Components of a Differentiated Services IP Network

The Diffserv approach to IP QoS seeks to achieve scalability by utilizing a packet marking scheme with the handling of traffic aggregates at the core. While the edges of the network may deal with per-flow state information, the core network devices only need to manage and maintain information on a limited number of classes. Edge network devices merge multiple flows into aggregated classes based on administrator specified policy. The following could be viewed as the major components of Diffserv Network:

- 1) Network Policy/Resource Management
- 2) Edge Device functionality

3)Core Device functionality

2.2.1 Network Policy/Resource Management

The Policy management infrastructure allows the network operator to specify QoS network policy- i.e. which traffic gets what kind of service on the network. This high level policy is translated into device-level and downloaded to individual routing devices. A policy management platform will consist of policy servers, policy editors, policy decision points and the policy enforcement points. Methods for distributing policy information in a cohesive consistent manner across the network are currently being discussed. Initially, policy management platforms will include some form of simple resource management.

Ideally, dynamic QoS request and allocation of service will be supported by this infrastructure. However, the area of dynamic network resource allocation is one that has many open issues e.g. since the state of the network is constantly changing, the timely distribution of this information to the policy managers is a major issue. Further, there is a requirement to ensure that this kind of control traffic does not consume more than its fair share of network resources. For now, initial realization of Differentiated Services QoS networks will rely on static provisioning and allocation of bandwidth.

2.2.2 Edge device functionality

Edge nodes are the second major component of a Diffserv network. These are typically Routers but can also be gateways, nodal bandwidth management devices and possibly trusted end-hosts. These nodes examine incoming packets and classify them according to the policy specified by the network operators. The packets are then marked (in the former TOS field) to reflect the desired level of service. Other functions that might be performed by the edge device include metering, traffic shaping or traffic policing often referred to as traffic conditioning. To ensure that user traffic profiles are adhered to, the edge device

responsibility also includes shaping or policing. This allows network operators to control the amount of traffic injected into a network and ensure some measure of end-to-end service differentiation. Traffic shaping is one possible method of restricting user traffic to its operator-defined profile. Traffic policing is the alternative to shaping. Policers are responsible for tracking incoming traffic against user profiles. Out-of-profile packets are either marked specially or discarded. Those packets that are specially marked can always be transmitted to downstream routers under normal conditions. Those routers can either transmit the packets onwards or discard them if they experience congestion or provisioning rules are violated.

2.2.3 Core device functionality

The final components in the Diffserv architecture are the Routers at the core of the network. These routers will have simpler functionality than the edge devices. These devices forward packets with certain PHB (Per Hop Behavior) depending on the particular packet marking. PHB is the Diffserv architecture term used to describe the forwarding treatment accorded to a packet at core network devices. It is the externally observable forwarding behavior applied against the packet at a particular hop. This observed behaviour is due to treatment received from a particular queuing discipline such as Priority Queuing, Weighted Fair Queuing, RED etc. Different packet markings will invoke different PHBs. Combinations of different PHBs and policy rules applied end-to-end result in a realized service as viewed by the user. Many methods may be used at the edge to decide on the packet marking required but the core network only has to inspect one fixed header field on a packet by packet basis in order to determine the forwarding treatment.

Network solutions for offering and guaranteeing QoS will vary according to the needs and requirements of each service provider. The above components are building blocks that will be combined to deliver some form of QoS relevant to particular networks. Dif-

Figure 2.1: Sample Network Configuration for DS

ferent networks will utilize only those building blocks that they require. Figure 2.1 illustrates a sample scenario where the various building blocks can be utilized to deliver end-to-end IP QoS. The Figure illustrates how two end-hosts would communicate together with their traffic going through a series of edge and core networking devices. In the depicted scenario, the end-host contracts a service with the local policy/resource manager. The Policy/Resource Manager translates this contract into device-relevant QoS traffic profile and downloads this to the edge device. Edge devices perform classification, marking and policing or shaping based on this policy. Core devices treat packets according to the PHB specified by the packet marking. All of this results in an end-to-end service as viewed by the users.

2.3 Type of Services On Diffserv Network

There have been a number of proposals describing service schemes that realize a Differentiated Services network. Two of the most popular proposals focus on Assured and Premium services.

A description of Premium Service can be found in [21]. This service is intended to provide a low delay, low jitter guarantee and is geared to the needs of real-time end-applications. The key elements in providing such a service include an EF (Expedited Forwarding) PHB and some form of strict shaping at the edge of the network. The EF PHB [5] ensures that packets are serviced at each node with bounded delay. The traffic shapers ensure that traffic is burst-limited and complies with the peak user profile. Packets exceeding the profile limit are either delayed or dropped. The packets are not forwarded by downgrading the service class marking.

The other key service proposal is Assured Service. A description of this kind of service can be found in [22][2]. The service is intended to allocate the bandwidth of the Internet to different users in a controlled way during periods of congestion. To alleviate congestion and limit delay in individual nodes, packets need to be dropped. The key to Assured Service is that packets marked with high priority are less likely to get dropped during congestion than packets marked with a lower priority. Thus, the higher priority packet is said to have a lower drop probability. The key elements in this service are the AF (Assured Forwarding) PHB [4] and some kind of policer at the edge of the network.

The policer is responsible for metering the flow and gauging whether or not it has exceeded its contracted traffic profile. If it is within its profile, it is marked with a higher priority. If it has exceeded its profile, intelligent marking is performed on the out-of-profile packets to indicate that downstream routers can treat them with higher drop probability during times of congestion. Intelligent marking is required for applications that utilize adaptive protocols such as TCP. This is because TCP's performance can be

greatly hindered by multiple sequential packet drops [23]. The AF PHB is responsible for enforcing differentiation when a router experiences congestion. It achieves this by mapping the packet markings to a suitable packet discard algorithm.

The AF PHB specifications are not intended to include details on how PHBs are to be implemented. It is left to the individual implementers of Diffserv devices to determine what underlying queuing mechanisms will be used to realize the PHBs. One possible implementation of the AF PHB is to use a single queue employing a RIO-like algorithm as proposed in [22]. Another possible implementation is to utilize multiple priority queues with multiple different levels of drop preference within each queue as hinted by [4]. Regardless of the specific implementation approach selected for the AF PHB, the packet discard algorithm will likely be based on the RED (Random Early Detect) buffer management technique [1]. The following sections describe different types of packet drop techniques (similar to RED) for queues with multiple drop preference within a queue.

2.4 MRED- Multi-Level RED

RED is an Active Queue Management (AQM) technique that detects incipient congestion and provides feedback to end hosts by dropping the packets. The motivation behind RED is to keep the queue size small, reduce burstiness and solve the problem of global synchronization [1]. RED drops packets before the actual physical queue is full. It operates based on an average queue length that is calculated using an exponential weighted average of the instantaneous queue length. RED drops packets with certain probability depending on the average length of the queue. The drop probability increases from 0 to maximum drop probability (max_p) as average queue size increases from minimum threshold (min_{th}) to maximum threshold (max_{th}). If average queue size goes above (max_{th}), all packets are dropped. Other variants of Active Queue Management with RED as their basic queue management are explained in the following sections.

2.4.1 Taxonomy

MRED is a generic term used to describe any scheme where drop probability for packets with different drop precedence (packet color) needs to be calculated independently for each drop precedence. The different drop precedence levels are also referred to in terms of colors as Green-DP0, Yellow-DP1 and Red-DP2. This is achieved by maintaining multiple sets of RED thresholds one for each drop precedence. In addition, the average queue used in the drop decision can be calculated using a number of different schemes. In [7], Goyal et al classify variants of RED into 4 categories. Figure 2.2 depicts the

Figure 2.2: Variants of RED

four categories with examples. SAST is simply RED [1] where a single set of RED parameters is maintained along with a single queue average. SAMT and MAMT are different ways of implementing MRED. The value of the MAST variant of MRED is not as intuitive as SAMT or MAMT and will not be considered further in this document. The author is not aware of any MRED scheme developed based on the MAST approach. SAMT is better known in the Internet community by the name WRED (Weighted RED). Two simple approaches to the MAMT variant of MRED are RIO-C (RED with In/Out and Coupled average Queues) and RIO-DC (RED with In/Out and Decoupled average Queues). Detailed descriptions of the differences and implications of these different

MRED variants are provided in the next section.

2.4.2 WRED - Weighted RED

Weighted RED calculates a single average queue that includes arriving packets of all colors [9]. For an arrival or departure of green, yellow or red packets, WRED updates a single average queue based on total number of packets of green, yellow or red color. However, multiple RED threshold parameters are maintained - one for each color. The parameters for packets of different colors can generally be set in three ways: (i) partially overlapped (Fig. 2.3a) (ii) overlapped (Fig. 2.3b) and (iii) staggered (Fig. 2.3c). Guidance for setting of WRED parameters can be found in [10]. In [10] partially overlapped (Fig. 2.3a) setting has been recommended most for WRED.

2.4.3 RIO-C (RED with In/Out and Coupled virtual Queues)

The traditional RIO [2] (RED with in/out) uses the same mechanism as in RED, but is configured with two sets of Parameters, one for IN packets and one for OUT packets. The packets of a traffic stream are marked in-profile (IN) when the stream is within the limits of specified policy. When the stream crosses the limits of the specified policy, its packets are marked as out-of-profile (OUT) packets. IN and OUT correspond to DP0 and DP1 or green and yellow packets. We refer to the RIO algorithm described in [2] as RIO-C. RIO-C is easily extended to three-drop precedence or colors [14]. We call RIO-C with three DP s as Generalized RIO-C. In GRIO-C, the average queue for packets of different colors is calculated differently. In such a scheme, average queue length of a color can be calculated by adding its average queue to the average queues of colors of lower drop precedence. For example, average queue size of green color packets will be calculated using green color packets only. For yellow color packets, the average queue will be calculated using green and yellow packets. For red packets, the average queue

a) Partially Overlapped

b) Overlapped

c) Staggered

Figure 2.3: Parameter setting options for MRED

will be calculated using red, yellow and green packets. GRIO-C derives its name from the coupled relationship of the average queue calculation.

The implication of this scheme is that the drop probability for packets with higher drop precedence is dependent on the buffer occupancy of packets having lower drop precedence. Thus, there is an inherent assumption that it is better to drop DP1 and DP2 packets for DP0 packets. Similarly, for DP1 packets, it is assumed better to drop DP2 packets. Services based on AF implementations utilizing GRIO-C will be subject

to this restriction. With GRIO-C, the parameters can be set using any of the methods outlined in Figure 2.3.

2.4.4 RIO-DC (RED with In/Out and De-Coupled Queues)

In RIO-DC (RIO with Decoupled Queues), the average queue for packets of each color is calculated independently. In this scheme, the average queue length for a color is calculated using number of packets of that color in the queue. The average queue length for green, yellow, and red packets will be calculated using the number of green, yellow and red packets in the queue respectively. The RED parameter settings can be chosen depending on the treatment to be given to different colors.

RIO-DC appears to be the most suitable MRED-variant if the operator desires to assign weights to different colored packets within the same queue i.e. there is no relationship between the packets marked green and yellow. Thus, the RIO-DC scheme could start to emulate the WRR scheduling scheme except that a single queue is used. Extensive testing for RED parameter configurations for this variant of MRED is reported in [11].

Chapter 3

Experimental Setup and Test Descriptions

Though the IETF has standardized the AF PHB and defined the PHB behavior, it is left to the implementers to develop the specific buffer management mechanism to realize the requirements of this PHB. Most implementations of AF will use RED or a similar Active Queue Management technique. The RED configuration where different sets of RED parameters are maintained for different colored packets is referred to as MRED (Multi-level RED). This work performs a study of two variants of MRED that can be used to meet the requirements of AF. We seek to understand the conditions for which the two variants produce similar results as well as the scenarios where the two different implementations yield different results. More details on the variants of MRED are contained in previous chapter.

Our experimental focus is on short-term bursty traffic that needs to be delivered without loss. This includes control traffic, which exhibits ON-OFF behavior, and transactional or short transfer traffic, which constitutes the majority of traffic on the Internet today. However the experiments were done for the Bulk transfer traffic also. The experimental testbed utilizes VxWorks-based implementation of Diffserv Edge devices and a

Linux Core Router running Linux-2.2.12.

3.1 Experimental Configuration

The experiments were done with two colors in order to isolate and study the specific buffer management algorithms: WRED and RIO-C. The study was performed using an experimental test-bed consisting of Diffserv edge and core devices. The experimental setup is depicted in Figure 3.1. The setup consists of three edge devices E1, E2, E3, a core device C1 and end-hosts, H1 - H4. The edge devices are based on a Pentium-II platform and utilized a VxWorks-based RTOS (Real Time Operating System) implementation of Diffserv. The core device is based on a Pentium II PC and utilized a Linux-2.2.12 implementation of Diffserv. Linux is chosen in the core as it is freely available and

Figure 3.1: Experimental Setup

results can be easily verified and further study can be pursued. The end hosts consisted of Pentium-II PC s running Linux-2.0.34 (TCP Reno stack) having as the operating

systems. There is a 10Mb link between edges and end hosts. The bottleneck link between C1 and E3 has a capacity of 1.7Mb. A configurable delay emulator device is used in the testbed to emulate network transmission delay. For the experiments, the one-way transmission delay is fixed at 30ms. In carrying out the experiments effort, were made to eliminate any non-homogeneity in hardware and software.

Edge devices in the testbed classify, police and mark packets based on source/ destination IP address. Token bucket SR2C (single rate two color) meter/policer is used for traffic conditioning [3]. Metering is used to monitor the data rate of policy traffic. The Policer based on metering information determines whether a packet is In-profile or Out-profile and classifies it accordingly. The SR2C marker uses two parameters Committed Information Rate (CIR) and a Token Bucket. Token Bucket is replenished at the rate of CIR. Packets that arrive when bucket has sufficient number of tokens are marked green else marked yellow.

At the core RIO-C and WRED were used for differentiated dropping of packets during congestion. Though the edge devices could mark packets with three colors, packets were marked with two colors only (however results obtained for two-color study can be extended for cases where three colors are used). NETPERF [13] and TCPBLAST tools were used during the experiment to generate TCP and UDP traffic from the end-hosts. Selection of RED parameter was followed from [11]. The authors report the results of studies with varied RED parameter settings. Different RED settings are shown in section 3.2 with the different test cases.

3.2 Experimental Descriptions and Results

This section describes the experiments and provides the results obtained. The experiments focus on the WRED and RIO-C schemes to understand their effectiveness in managing the buffer under different conditions. For each of the schemes, three sets of

Tests Done	RED		
	Staggered Fig. 2.3c (Table 3.2)	Partially Overlapped Fig. 2.3a (Table 3.3)	Overlapped Fig. 2.3b (Table 3.4)
On-Off traffic	Staggered Fig. 2.3c (Table 3.2)	Partially Overlapped Fig. 2.3a (Table 3.3)	Overlapped Fig. 2.3b (Table 3.4)
Transactional transfers	Staggered Fig. 2.3c (Table 3.5)	Partially Overlapped Fig. 2.3a (Table 3.6)	Overlapped Fig. 2.3b (Table 3.7)
Bulk transfers	Staggered Fig. 2.3c (Table 3.8)	Partially Overlapped Fig. 2.3a (Table 3.9)	Overlapped Fig. 2.3b (Table 3.10)

Table 3.1: Tests Performed

experiments were performed with three different types of Internet traffic: (i) ON-OFF bursty traffic to replicate control traffic (ii) transactional transfers to replicate web traffic and (iii) bulk transfers. All the tests are done for ten minutes duration. Each set of experiments was done using staggered, partially overlapped and fully-overlapped RED parameter settings (Fig. 2.3). The experiments were repeated sufficient number of times to gain confidence in results.

3.2.1 Experiment 1: ON-OFF bursty traffic

This section reports results of experimentation when aggregated traffic arrives at an AF queue in ON-OFF fashion. The tests were repeated using the three sets of RED parameter models shown in Figure 2.3.

a) Staggered RED

Staggered RED parameter settings shown in Figure 3.2 were used. Continuous UDP traffic at the rate of 9Mb/s was sent between H2 and H4 as background traffic to create congestion at the core. The entire traffic stream was marked as yellow (DP1). From H1 to H3, the traffic consisted of short duration UDP bursts with 120ms ON time and 2s OFF time. All of the traffic between H1 and H3 was marked as green (DP0). A fixed

Figure 3.2: Staggered RED parameter setting

packet size of 1024bytes was used for all traffic. The total number of packet drops was observed at the core. Table 3.2 summarizes the results. As seen from the table, RIO-C completely protects the bursty ON-OFF traffic even in the presence of congestion. For RIO-C, the DP0 queue is empty in the absence of any green traffic but the DP1 queue average operates around max_{th} (Fig. 3.3). When a small burst of green packets arrives, the DP0 average queue still remains below min_{th} . However, incoming yellow packets

Figure 3.3: Queue variations for ON-OFF traffic: RIO-C

Buffer management scheme	Marking policy	Total Packets rx'd(core)	Dropped packets	% of total dropped	Early packet drops(%)	Forced packet drops(%)	Observation
RIO-C							
ON-OFF CBR-UDP	DP0(Green) DP1(Yellow)	42000 750000	0 656504	0 87.53%	0 6708 (0.89%)	0 649788 (86.6%)	Green pkts totally protected
WRED							
ON-OFF CBR-UDP	DP0(Green) DP1(Yellow)	42000 750000	2878 655720	6.85% 87.43%	485 (1.15%) 10363 (1.38%)	2393 (5.7%) 645357 (86.05%)	Green pkts unprotected

Table 3.2: ON-OFF bursty traffic- Staggered RED

Figure 3.4: Queue variations for ON-OFF traffic: WRED

cause the DP1 average to go above max_{th} , so the yellow packet are going to be dropped by WRED. The queue coupling in RIO-C prevents any green packets to be dropped at the expense of yellow packets i.e. all yellow packets must be dropped before any green packet is dropped. In the case of WRED, this is not the case. Since there is a single average queue, even in the absence of green traffic, the average queue hovers around max_{th} of DP1 i.e. min_{th} of DP0 (Fig. 3.4). So when bursts of green traffic arrive, the DP0 queue average rises to a level between its min_{th} and max_{th} , thereby dropping some green

packets. From Table 2 we observe that WRED dropped 6.85% of the ON-OFF traffic while RIO-C did not drop any of the ON-OFF traffic.

b) Partially Overlapped RED

Experiment in section a) was repeated with partially overlapped (Fig. 2a) RED settings as depicted in Figure 3.5. These settings are similar to those suggested in [10]. Con-

Figure 3.5: Partially Overlapped RED parameter settings

gestion creating UDP traffic was reduced to 2Mb/s. UDP traffic at 9Mb/s is found to generate heavy drops for both RIO-C as well as WRED under these RED settings hence fair evaluation is not possible. Table 3.3 summarizes the results.

Tables 3.2 and 3.3, we observe that under this new choice of RED parameter configuration, WRED performance in terms of protecting DP0 packets has decreased. In this test, almost 10.5% of ON-OFF traffic were dropped. It is because of increase of *max_u* for DP1 i.e. fewer forced drops of DP1 packets, cause more of the available buffer to be consumed by yellow packets. This causes an increase in DP0 packet drops. In contrast, RIO-C is still able to protect DP0 packets as evidenced by the fact that it does not drop any such packets.

Buffer management scheme	Marking policy	Total Packets rx'd(core)	Dropped packets	% of total dropped	Early packet drops(%)	Forced packet drops(%)	Observation
RIO-C							
ON-OFF CBR-UDP	DP0(Green) DP1(Yellow)	42000 165000	0 70492	0 42.72%	0 9050 (5.48%)	0 61442 (37.24%)	Green pkts totally protected
WRED							
ON-OFF CBR-UDP	DP0(Green) DP1(Yellow)	42000 165000	4394 77272	10.46% 46.83%	626 13195 (1.5%) (8.0%)	3768 64077 (8.97%) (38.83%)	Green pkts unprotected

Table 3.3: ON-OFF bursty traffic- Partially overlapped RED

c) Overlapped RED

The third test for this type of ON-OFF traffic used RED parameters shown in Figure 3.6. Above Experiment 3.2.1b was repeated with these new RED parameters. Table 3.4 shows the observed results at the core device. Examining the results for WRED, we observe that the number of dropped DP0 packets has increased from 10% in the previous ex-

Figure 3.6: Overlapped RED parameter settings

periment to 12.5% in this test. This is because the increase of min_{th} for DP1 packets provides more protection to yellow packets as compared to previous experiments, thereby

Buffer management scheme	Marking policy	Total Packets rx'd(core)	Dropped packets	% of total dropped	Early packet drops(%)	Forced packet drops(%)	Observation
RIO-C							
ON-OFF CBR-UDP	DP0(Green) DP1(Yellow)	42000 165000	0 70508	0 42.73%	0 5702 (3.45%)	0 64806 (39.28%)	Green pkts totally protected
WRED							
ON-OFF CBR-UDP	DP0(Green) DP1(Yellow)	42000 165000	5254 75887	12.5% 46.0%	646 16188 (3.45%) (9.81%)	4608 59699 (8.97%) (36.18%)	Green pkts unprotected

Table 3.4: ON-OFF bursty traffic: Overlapped RED

increasing more dropped green packets.

3.2.2 Experiment 2: Transactional transfers

The experiments with transactional transfers were done to imitate web transfers and short query-response transactions, which constitute majority of the Internet traffic [17]. Netperf's TCP_CRR (connect request/response) was used to imitate transactional transfers. TCP_CRR opens a TCP connection. It sends a request of a given size and tears down the connection after receiving the response of prescribed size. A new connection is started immediately with zero delay. Request size of 1KB and response size of 5KB was used for transactions. The tests were done using the three different sets of RED parameter used in previous tests.

a) Staggered RED

The RED parameters are set as given in Figure 3.2. Background traffic CBR UDP traffic at 9Mb/s was sent between H2 and H4, all marked yellow. Transactions were done with H3 sending a request and H1 giving a response. Forty TCP_CRR flows were started with

Buffer management scheme	Marking policy	Total Packets rx'd (core)	Dropped packets	% of total dropped	Early packet drops (%)	Forced packet drops (%)	Transaction /sec	No. of Retx-mn's	Observation
RIO-C									
Trans. Xfer CBR-UDP+ Excess trans. Xfer	DP0(Green) DP1(Yellow)	85734 751158	0 668645	0 89.0%	0 7332 (0.97%)	0 661313 (88.03%)	0.52	0 756	Green pkts totally protected
WRED									
Trans. Xfer CBR-UDP+ Excess trans. Xfer	DP0(Green) DP1(Yellow)	75095 750843	56 660738	0.07% 88.0%	56 8827 (0.97%)	0 651911 (88.03%)	0.45	993	Green pkts not protected & low transact. rate

Table 3.5: Transactional transfers: Staggered RED parameters

zero delay between the start times. The TCP_CRR traffic was marked green as long as it conformed to target rate of 1.7Mb/s. The number of new TCP connections and packet retransmissions during the test interval were observed at end hosts.

Table 3.5 provides the results. From this table, we observe that with RIO-C, the TCP_CRR was able to complete 304 transactions, while with WRED, 267 transactions were completed during the same ten minute test period. In addition, while the WRED test showed some packet drops, there were no packet drops during the tests with RIO-C. The 13% downfall in transaction rate of WRED can be attributed to dropping of TCP_CRR packets marked as green and yellow as shown by number of retransmissions. Bagal et al [20] has shown that even with RED in routers in the end to end path, TCP RENO is susceptible to multiple drops within same window. RENO is known to overreact to such a case affecting overall performance. Bagal et al [20] has shown that RENO is still most widely deployed TCP.

b) Partially Overlapped RED

In this test, the RED settings are changed to use the partially overlapped model as shown

Buffer management scheme	Marking policy	Total Packets rx'd (core)	Dropped packets	% of total dropped	Early packet drops (%)	Forced packet drops (%)	Transaction /sec	No.of Retx-mn's	Observation
RIO-C									
Trans. Xfer	DP0(Green)	59382	0	0	0	0	0.39	0	Green pkts totally protected
CBR-UDP+ Excess trans. Xfer	DP1(Yellow)	166434	70283	42.2%	15034 (9.03%)	55249 (33.19%)		580	
WRED									
Trans. Xfer	DP0(Green)	34233	6062	17.17%	393 (1.15%)	5669 (16.55%)	0.16	5039	Green pkts not protected&much low transact. rate
CBR-UDP+ Excess trans. Xfer	DP1(Yellow)	165011	46815	28.37%	18728 (11.3%)	28087 (17.02%)			

Table 3.6: Transactional transfers: Partially Overlapped RED parameters

in Figure 3.5. The background UDP CBR traffic between H2 and H4 is sent at the rate of 2Mb/s, all of which is marked yellow. Above Experiment 3.2.2a is then repeated under these new network conditions. The results in Table 3.6 show that the total number of transactions for both the WRED and RIO-C is reduced as compared against the results in Table 3.5.

Again, the scenario utilizing RIO-C provides a higher transaction rate than the scenario with the WRED scheme. In comparison with Table 3.5, we observe that the WRED performance has degraded as compared with RIO-C. This is due to large number of packet drops at the core. In this case, the queue size for the WRED is observed to operate near *max*. Because there is a single average queue, we observe drops of DP0 packets, thus affecting TCP flows. In the case of RIO-C, its transaction rate is reduced from the results of above experiment because the DP1 packets are allowed a greater portion of the buffer. However, since it maintains two distinct queue averages, it nevertheless protects DP0 packets from being dropped. RIO-C has some retransmissions due to dropping of DP1 packets(Xactional Xfer packets marked as yellow).

Buffer management scheme	Marking policy	Total Packets rx'd (core)	Dropped packets	% of total dropped	Early packet drops (%)	Forced packet drops (%)	Transaction /sec	No.of Retx-mn's	Observation
RIO-C									
Trans. Xfer	DP0(Green)	66187	0	0	0	0	0.405	0	Green pkts totally protected
CBR-UDP+ Excess trans. Xfer	DP1(Yellow)	166399	75058	45.1%	18071 (10.86%)	56987 (34.24%)		680	
WRED									
Trans. Xfer	DP0(Green)	38282	6648	17.36%	484 (1.26%)	6164 (16.1%)	0.17	5645	Green pkts not protected&much low transact. rate
CBR-UDP+ Excess trans. Xfer	DP1(Yellow)	165007	49158	29.8%	22977 (13.92%)	26181 (15.87%)			

Table 3.7: Transactional transfers: Overlapped RED

c) Overlapped RED

In this experiment, the RED settings are changed to the overlapped RED model with parameters as shown in Figure 3.6. We repeat the above experiment 3.2.2b. The observed results can be found in Table 3.7. RIO-C and WRED performance appears to improve slightly over the test results in 3.2.2b. Due to the increase of min_w , we observe a greater number of DP1 packets in the queue. As RED drop packets on the basis of bandwidth occupied by a flow, it results in increase in earlier drops thereby increasing the overall drop percentage of DP1 packets. This results in some improvement over partial RED settings.

3.2.3 Experiment 3: Bulk transfers

Experiments with bulk data transfer were also performed to understand the behavior of RIO-C and WRED with long-lived TCP flows such as FTP. The test was performed for ten minutes and repeated for all three RED parameter model settings. All the UDP traffic is marked DP1 as in earlier tests. Netperf's TCP_RR (request/response) is used for generating continuous TCP flows between H1 and H3. TCP_RR opens one connection

for the whole test period. It then keeps on generating requests and responses of the prescribed size without any delays. Request size of 64KB and response size of zero byte i.e. no response, were used. Three TCP flows are started simultaneously. The three flows share the target rate of 1.7Mb/s thus, causing green packets to arrive at the buffer at this rate.

Setting a target rate that is the same as the bottleneck link bandwidth (1.7Mbps) essentially mimics a network that is running at full capacity. Previous studies [12] have shown that such networks have difficulty assuring the target rate. However, we choose this target rate in order to study what happens at a core device where aggregated traffic may arrive in the form of bursts of yellow and green traffic. This is important to understand because even though network engineers will strive to over-provision Diffserv networks, in the absence of route pinning or MPLS, there may be periods where particular core devices receive more traffic than they are configured for. This will then cause buffers to overflow or operate at capacity. It is precisely this kind of scenario that we seek to study in this testcase.

a) Staggered RED

This experiment uses the RED settings of Figure 3.2. UDP CBR traffic is sent at 9Mb/s from H2 to H4. Throughput, packet retransmissions, packet drops were observed to compare the performance. Table 3.8 provides the results.

RIO-C and WRED are found to give comparable performance under these settings. RIO-C is also unable to achieve its target rate. This can be attributed to some packet drops of TCP flows marked as DP1 and due to high transmission rate of DP1 UDP traffic, even a slight backing off by TCP traffic is rapidly taken by it. The large numbers of retransmission are accounted by the large percentage dropping of DP1 packets of TCP test flows.

Buffer management scheme	Marking policy	Total Packets rx'd (core)	Dropped packets	% of total dropped	Early packet drops (%)	Forced packet drops (%)	Throughput (Mb/s)	No.of Retx-mn's	Observation
RIO-C									
Bulk	DP0(Green)	77113	0	0	0	0	1.5	0	Green pkts totally protected
CBR-UDP+ Excess bulk	DP1(Yellow)	750683	716140	95.4%	1993 (0.26%)	714147 (95.14%)		861	
WRED									
Bulk	DP0(Green)	67709	51	0.07%	51	(0.07%)	1.3		Green pkts not protected& lower thro'put
CBR-UDP+ Excess Bulk	DP1(Yellow)	750720	698066	93%	4628 (0.62%)	693438 (92.3%)		896	

Table 3.8: Bulk Transfers: Staggered RED parameters

b) Partially Overlapped RED

With partial overlapped RED settings same as in earlier test cases, background UDP traffic is similarly kept at 2Mb/s. Table 3.9 gives the observed readings. Performance

Buffer management scheme	Marking policy	Total Packets rx'd (core)	Dropped packets	% of total dropped	Early packet drops (%)	Forced packet drops (%)	Throughput (Mb/s)	No.of Retx-mn's	Observation
RIO-C									
Bulk	DP0(Green)	55328	364	0.65%	364 (0.65%)	0	1.1	378	Small% of Green pkts not protected
CBR-UDP+	DP1(Yellow)	165010	103934	63%	9350 (5.66%)	94584 (57.32%)			
WRED									
Bulk	DP0(Green)	8371	1008	12.04%	86 (1.02%)	922 (11.01%)	0.15	1138	Green pkts not protected & lower thro'put
CBR-UDP+	DP1(Yellow)	165000	38457	23.3%	20382 (12.35%)	18075 (10.95%)			

Table 3.9: Bulk Transfer: Partially Overlapped RED parameter

of both RIO-C and WRED deteriorates. RIO-C performance is affected by increase in packet drops. However it vastly outperforms WRED. In case of WRED, DP1 queue is observed to operate around *max* i.e. 50KB, when DP0 packets arrive most likely

the queue size will be in RED region and hence it can be dropped with certain drop probability. If queue size goes beyond 50KB, the packet will be forcibly dropped. From Table 3.9, it can be seen that high percentage of packets of TCP test flows is dropped. As a result TCP flows never gets a chance to fully open its window, thereby affecting the throughput. Due to unresponsive UDP flow in the background it takes Full usage of the available bandwidth. As can be seen from the table also, packet drops for DP1 in case WRED are far less than RIO-C, which protect DP0 packets at the expense of DP1 packets.

c) Overlapped RED

Experiment 3.2.3b is repeated with overlapped RED settings of Fig 3.6. Table 3.10 gives the observed values. Performance of WRED improves slightly from partially overlapped test case. This is similarly explained as in transactional transfer test case 3.2.2c.

Buffer mngmn't scheme	Marking policy	Total Packets rx'd (core)	Drop'd packets	% of total drop'd	Early packet drops (%)	Forced packet drops (%)	Throughput (Mb/s)	No.of Retx-mn's	Observation
RIO-C									
Bulk	DP0(Green)	54744	357	0.65%	357 (0.65%)	0	1.1	377	Small% of Green pkts not protected
CBR-UDP	DP1(Yellow)	165035	103162	62.5%	12177 (7.38%)	90985 (55.13%)			protected
WRED									
Bulk	DP0(Green)	10008	1010	10.1%	123 (1.23%)	887 (8.86%)	0.2	1132	Green pkts not protected
CBR-UDP	DP1(Yellow)	165000	40638	24.63%	25305 (15.33%)	15333 (9.29%)			lower thro'put

Table 3.10: Bulk transfers: Overlapped RED parameters

3.2.4 Experiment 4: Varying RED parameters for WRED

In [6], the authors have concluded that for Differentiated Services Assured Forwarding based services, similar performance can be achieved with either a SAMT or MAMT scheme via proper selection of RED parameters. For selected RED settings, we have not found this to be the case for our experiments. For the experiments with ON-OFF traffic, Figure 3.2 settings provided the best results for WRED. This is because the RED model utilized (staggered) allowed a greater degree of protection of DP0 packets. However, even this model was insufficient to completely protect DP0 packets for WRED. We decided to experiment with grossly staggered RED parameters i.e. increase min_{th} for DP0 well beyond max_{th} of DP1 to see at which point, WRED performed comparably to RIO-C.

The test case with ON-OFF traffic is chosen. Thus, experiment 3.2.1a was repeated and RED parameters ($\{emph{min_{th}}, max_{th}\}$) were varied for DP0 traffic till WRED appreciably protects the DP0 packets. The same $maxp$ and Wq parameters were used as in all other tests. Table 3.11 illustrates the results. For the protection of DP0 packets for ON-OFF traffic, we observe from table that WRED is able to achieve the same level of performance as RIO-C, if we move the thresholds up at least by 25KB. Thus WRED will incur a clear cost in terms of increased latency and increased memory requirement if thresholds are to be increased by 25KB.

3.3 Conclusions

Figures 3.7, 3.8 and 3.9 provide a graphical representation of results. Examining the results and figures yield the following observations.

1. For ON-OFF bursty traffic, RIO-C is better than WRED in protecting DP0 traffic under all RED settings (Fig. 3.7a). WRED gives its best performance under

Test performed	<i>Min_{th}</i> (KB)	<i>Max_{th}</i> (KB)	Marking Policy rx'd(core)	Total Packets	Dropped packets	% of total
ON-OFF	30	55	DP0(Green)	42000	714	1.7%
CBR-UDP	10	25	DP1(Yellow)	750000	659898	88.1%
ON-OFF	40	65	DP0(Green)	42000	85	0.2%
CBR-UDP	10	25	DP1(Yellow)	750000	660811	88.1%
ON-OFF	40	70	DP0(Green)	42000	36	0.08%
CBR-UDP	10	25	DP1(Yellow)	750000	660762	88%
ON-OFF	50	75	DP0(Green)	42000	4	
CBR-UDP	10	25	DP1(Yellow)	750000	660913	88.12%

Table 3.11: WRED with Grossly Staggered Parameters

staggered RED settings.

2. RIO-C outperforms WRED in transactional transfers also with better protection and higher transactional rate for DP0 traffic (Fig. 3.7b & 3.8). Staggered RED settings again turn out to be giving the best performance for both buffer management schemes.

3. In case of Bulk transfers, RIO-C's protection of DP0 traffic in case of Bulk transfers, RIO-C's protection of DP0 traffic overlapped RED settings (Fig. 3.7c). It still vastly outperforms WRED in terms of throughput (Fig. 3.9). Staggered RED settings appear to be the optimal configuration in this test case also.

4. As observed for ON-OFF traffic, for WRED to obtain same results as RIO-C, WRED needs to be grossly staggered (Table 3.11). This improvement in WRED performance with different RED parameters has to be considered vis-a-vis buffer availability, acceptable latency and type of assured service required. For protection of higher drop precedence traffic at the cost of lower DP traffic, RIO-C is definitely a better alternative.

a) Packet drops - ON-OFF tests

b) Packet drops - Transactional Xfer tests

c) Packet drops - Bulk transfer tests

Figure 3.7a,b,c: Packet drops for RIO-C & WRED for all three tests

Figure 3.8: Transaction rate - Transactional Xfer tests

Figure 3.9: Throughput - Bulk Transfer tests

3.4 Future Work

In the previous sections we evaluated two Buffer Management schemes i.e. WRED and RIO-C for Differentiated Services Assured Forwarding PHB. RIO-C is found to outperform WRED under considered test cases. The study was done with two Drop Precedence traffic in a single AF class. This study of WRED and RIO-C can be extended further with three Drop Precedence traffic in a single AF class. Evaluation of RIO-C and WRED, coupled with some scheduling mechanism, for all four AF classes can be another area for detailed study.

Explicit Congestion Notification based congestion mechanism marks the packet instead of dropping it during times of congestion. Marking high priority i.e. low Drop Precedence traffic as ECN capable and then evaluating WRED and RIO-C can be another direction for future research work.

Chapter 4

Congestion Control in TCP

4.1 Introduction

In current IP networks, congestion management is left to the protocols running on top of IP. A router (running IP) when congested simply drops packets. TCP is the most dominant transport protocol today [17]. TCP infers that there is congestion in the network by detecting packet drops (RFC 2001). Congestion control algorithms [23] are then invoked to alleviate congestion. TCP initially sends at a higher rate (slow start) until it detects a packet loss. A packet loss is inferred by the receipt of 3 duplicate ACKs or detected by a timeout. The sending TCP then moves into a congestion avoidance state where it carefully probes the network by sending at a slower rate (which goes up until another packet loss is detected).

Traditionally a router reacts to congestion by dropping the packet in the absence of buffer space. This is referred to as Tail Drop. This method has got a number of drawbacks (outlined in Section 2). These drawbacks coupled with the limitations of end-to-end congestion control have led to interest in introducing smarter congestion control mechanisms in routers.

One such mechanism is Random Early Detection (RED) [1] which detects incipient

congestion and implicitly signals the over-subscribing flow to slow down by dropping its packets. A RED-enabled router detects congestion before it actually occurs based on a running average queue size and drops packets probabilistically before the queue actually fills up. The probability of dropping a new arriving packet increases as the average queue size increases above a low water mark (min_{th}), towards higher water mark (max_{th}). Above max_{th} all packets are dropped.

An extension to RED is to mark the IP header instead of dropping packets (when the average queue size is between min_{th} and max_{th} : above max_{th} the packets are dropped as before). Cooperating end systems would then use this as a signal that the network is congested and slow down. This is known as Explicit Congestion Notification (ECN).

Another alternative approach to current ECN mechanism is Backward Explicit Congestion Notification (BECN). It involves the use of marking packets like ECN and use of existing IP signaling mechanism Internet Control Message Protocol (ICMP) Source Quench messages. It is suggested by Hadi et al in their draft [30]. This work studies comparative evaluation of ECN and BECN.

4.2 Queue management in routers

TCP's congestion control and avoidance algorithm are necessary and powerful but are not enough to provide good service in all circumstances since they treat the network as a black box. Some sort of control is required from the routers to complement the end system congestion control mechanisms. More detailed analysis is contained in [31]. Queue management algorithms traditionally manage the length of packet queues in the router by dropping packets when congestion occurs. A maximum length for each queue is configured. The router will accept packets till this maximum size is exceeded, at which point it will drop incoming packets. New packets are accepted when buffer space allows. This technique is known as *Tail Drop*. This method has served the Internet well for

years, but has the several drawbacks.

Since all arriving packets from all flows are dropped after the maximum threshold, this interacts badly with the congestion control mechanism of TCP. A cycle is formed with burst drops after the maximum queue size is exceeded, followed by a period of underutilization at the router as end systems back off. End systems then increase their windows simultaneously up to a point where a burst drop happens again. This phenomenon is called *Global Synchronization*. It leads to poor link utilization and lower overall throughput [31].

4.2.1 Active Queue Management

Active queue management mechanisms detect congestion before the queue overflows and provide an indication of this congestion to the end nodes [29]. With this approach TCP does not have to rely only on buffer overflow as the indication of congestion since notification happens before serious congestion occurs. One such active management technique is RED.

4.2.2 Random Early Detection

Random Early Detection (RED) [1] is a congestion avoidance mechanism implemented in routers which works on the basis of active queue management. RED addresses the shortcomings of Tail Drop. A RED router signals incipient congestion to TCP by dropping packets probabilistically before the queue runs out of buffer space. This drop probability is dependent on a running average queue size to avoid any bias against bursty traffic. A RED router randomly drops packets, such that the probability of dropping a packet belonging to a particular flow is approximately proportional to the flow's share of bandwidth. The rule is, if the sender is using relatively more bandwidth it gets penalized by having more of its packets dropped.

RED operates by maintaining two levels of thresholds minimum (min_{th}) and maximum (max_{th}). It drops a packet probabilistically if and only if the average queue size lies between the min_{th} and max_{th} thresholds. If the average queue size is above the maximum threshold, the packet is always dropped. When the average queue size is between the minimum and the maximum threshold, each arriving packet is dropped with probability $maxP$, where $maxP$ is a function of the average queue size. As the average queue length varies between min_{th} and max_{th} , drop probability increases linearly towards a configured maximum drop probability $maxP$. Beyond max_{th} , the drop probability is 100%. Figure 4.1 shows this relationship. Dropping packets in this way assures that when some subset of

Figure 4.1: Figurative description of RED

the source TCP packets get dropped they invoke their congestion avoidance algorithms that will ease the congestion at the gateway. Since the dropping is distributed across flows, the problem of global synchronization is avoided.

4.3 Explicit Congestion Notification

Explicit Congestion Notification is an extension proposed to RED which marks a packet instead of dropping it when the average queue size is between min_{th} and max_{th} [29]. Since ECN marks packets before congestion actually occurs, this is useful for protocols

like TCP that are sensitive to even a single packet loss. Upon receipt of a congestion marked packet, the receiver informs the sender (in the subsequent ACK) about incipient congestion which will in turn trigger congestion avoidance algorithm at the sender. ECN requires support from both the router as well as the end hosts, i.e. the end hosts TCP code needs to be modified. Flows that are not ECN capable will continue to be dropped by RED.

4.3.1 Implementations issues for ECN

Router side support for ECN can be added by modifying current RED implementations. For ECN capable flows the router marks the packets rather than dropping them (if the average queue size is between min_{th} and max_{th}).

The ECN Capable Transport (ECT) bit is set by the sender end system in IP header if both the end systems are ECN capable. Packets encountering congestion are marked using the Congestion experienced (CE) bit in the IP header by the router (if the average queue size is between min_{th} and max_{th}) on their way to the receiver end system (from the sender end system), with a probability proportional to their bandwidth usage following the procedure used in RED (RFC2309) routers. Bits 6 and 7 of the IPV4 header Service Type field are proposed as the ECT and CE place holders.

At the host side the proposal to add ECN specifies two new flags in the reserved field of the TCP header. Bit 9 in the reserved field of the TCP header is designated as the ECN-Echo (ECE) flag and Bit 8 is designated as the Congestion Window Reduced (CWR) flag.

4.3.2 Functional Description of ECN

The following points describe the connection set up for ECN capable flows, as well as the subsequent behavior of TCP for ECN flows, in case there is congestion experienced

in the network.

1. In the connection setup phase, the source and the destination TCPs have to exchange information about their desire and/or capability to use ECN. This is done by setting both the ECN-Echo flag and the CWR flag in the SYN packet of the initial connection phase by the sender; on receipt of this SYN packet, the receiver will set the ECN-Echo flag in the SYN-ACK response. Once this agreement has been reached, the sender will thereon set the ECT bit in the IP header of data packets for that flow, to indicate to the network that it is capable and willing to participate in ECN.
2. When a router experiences congestion it sets the CE bit in the IP header if the IP-ECT bit is set. When such a packet reaches the receiver, it responds by setting the ECN-Echo flag (in TCP header) in the next outgoing ACK for the flow. It will continue to do this in subsequent ACKs till it receives from the sender an indication that it (the sender) has responded to the congestion notification.
3. Upon receipt of this ACK, the sender triggers its congestion avoidance algorithm by halving its congestion window (*cwnd*) and updating its congestion window threshold value (*ssthresh*). Once it has taken these appropriate steps, the sender sets the CWR bit on the next outgoing packet to tell the receiver that it has reacted to the receiver's notification of congestion. The receiver reacts to the CWR by halting the sending of the congestion notifications (ECE) to the sender if there is no new congestion in the network.

Figure 4.2 describes ECN once the connection is established and both sides are ECN capable. The sender reaction to the indication of congestion in the network (when it receives an ACK packet that has the ECN-Echo flag set) is equivalent to the Fast Retransmit/Recovery algorithm (when there is a congestion loss) in non-ECN-capable

TCP i.e. the sender halves the congestion window *cwnd* and reduces the slow start threshold *ssthresh*. The Fast Retransmit/Recovery is still available for ECN capable stacks.

Explicit Congestion Notification

Figure 4.2: Pictorial representation of ECN

4.4 Backward Explicit Congestion Notification

Backward Explicit Congestion Notification (BECN) is an alternative approach to the current ECN mechanism involving the use of an existing IP signaling mechanism, the Internet Control Message Protocol (ICMP) Source Quench message. It has been suggested by Hadi et al in [30]. The use of ICMP Source Quench (ISQ) allows a basic ECN mechanism for IP, which does not require any negotiation between end systems. Congestion

notification is kept at the network (IP) level. The congestion state can be reflected up to the transport layer (e.g. TCP or UDP) for appropriate action. The authors in their draft believe that the ISQ based approach reduces the reaction time to congestion in the network.

In addition, they have proposed a mechanism in which the ISQ message can include information on the severity of congestion, allowing the end host to react accordingly so as to make maximal use of the resources and thereby maximizing network utilization. It is termed as MECN (Multiple Explicit Congestion Notification). In MECN, the ISQ message sent back to the sender will include the information of the severity of congestion and the sender, based on this information will adjust its *cwnd*. Further details can be found in [30]. In this thesis we limit ourselves to BECN, its implementation and evaluation. IP currently does not have any adhered mechanism to notify its transport protocols of network congestion problems. ISQ's have been used in the past for congestion notification. TCP implements its own congestion control algorithm to infer congestion as explained earlier.

4.4.1 Drawbacks of ECN

In the previous section issues related with ECN were described. Following are some of the drawbacks that are intuitive in case of ECN.

1. ECN is coupled to the transport layer (TCP) via the use of header information (ECE) bit. To extend this proposal to other transport protocols will require changes to each of their respective headers. BECN proposes to decouple the transport level protocols and use the source quench mechanisms to notify the transport layer of congestion problems. This provides the value that all IP transport protocols are notified in the same manner about network congestion. Here we will be considering the effects of BECN for TCP only.

2. ECN requires the congestion notification to incur a round trip time (RTT) before the sender can react (since the sender has to wait for the acknowledgment of a packet before it gets notified of incipient congestion). In a path with high delay bandwidth product this would be problematic for two reasons: i) In the scenario where the delay-bandwidth product is dominated mostly by high bandwidth (as in high speed networks), a large amount of traffic will pass through the intermediate routers causing an increase in congestion level before the sender is notified. ii) In the scenario where the delay-bandwidth product is dominated mostly by high latency/RTT (as in satellite networks), the reaction will take too long to address the congestion. In both cases, the efficient use of available bandwidth is affected. BECN avoids this by having the router send the congestion notification as soon as congestion occurs in the router to the sender.
3. Because of the binary nature of the feedback of ECN, the reaction is limited to halving the window size even if the congestion level is very low. Network resources could be more effectively utilized if the feedback was indicative of the congestion level at the overloaded point in the network. This is explained in the draft [30] under MECN.

4.4.2 Implementation issue for BECN

Before explaining the actual BECN algorithm we digress here to provide a brief history of why use of ISQ's for congestion notification was discouraged and arguments put forth by the proponents of BECN for the use of ISQ's.

- Router CPU inefficient use while processing these extra messages.
- The authors argue that CPU time is no longer a constrained resource today. It has been shown [1] that when using RED (with cooperating end systems) less packet

drops happen at the router in comparison to the traditional drop-tail algorithms used in disapproving ISQ. This implies the amount of processing needed at the router is reduced. Further faster reaction to congestion is provided but ISQ's would alleviate it faster, resulting in further reductions.

- Bandwidth consumption on reverse path.
- ISQ's are supposed to consume too much bandwidth in the reverse path because of tail drop i.e. once the maximum queue capacity of queue is reached all incoming packets will be dropped , thus generating a lot of ISQ's. However due to RED being a part in BECN, ISQ's will be generated at a rate proportional to the connection's share of the bandwidth at the congested router. Given this scheme, which addresses congested routers sequentially on a downstream path, it can be argued that the back-path even if it is the same as the forward path, is probably not really congested since it covers the path only to the first point of congestion along that path.

4.4.3 BECN Implementation- Router

A BECN capable router is assumed to have an active queue management algorithm implemented in it. Taking RED as an active queue management present in the router, following modifications are required.

1. If the incoming packet causes the average queue size to go above the maximum threshold, drop the packet.

If the ECT bit is set in the IP header (specifying that end host is BECN capable), send as ISQ back to the source.

2. If the incoming packet causes the average queue to go between the minimum and maximum, and if RED probability chooses this packet,

if the ECT bit is set and if the packet is not already marked(i.e. the CE bit is not set) then mark the packet (i.e. set the CE bit) and send back an ISQ
else drop the arrived packet.

4.4.4 BECN Implementation-Source End Hosts

If an ISQ message is received then the sender knows that there is network congestion. The flow causing the congestion is identified from the ICMP data. The TCP source reacts by halving both the congestion window and the slow start threshold value for that flow.

The sender does not react to ISQ's more than once per window. Upon the receipt of an ISQ packet at time t , it notes the packets that are outstanding at that time (sent but not yet acked) and waits until a time when they have been acknowledged before reacting to a new ISQ message. Figure 4.3 is a pictorial representation of the way BECN works. Following is the sequence of events in case the router was ECN or BECN capable.

1. The packet leaves the source to the destination and is first intercepted by the router R1.
2. If the router is congested than depending whether the router is ECN or BECN the following will happen
If it is ECN mark the packet
If it is BECN mark the packet and send an ICMP source quench to the sender -(2)
which in turn will react by slowing down its packet sending rate.
3. The packet is sent to the intermediate router R2-(3)
4. It reaches the end system (4) and the end system will send an ACK for the packet-(5).

Figure 4.3: Pictorial representation of ECN and BECN

If the packet was marked in case of ECN, the ACK will carry the information that the flow is causing congestion by setting ECN-echo bit. This will go back to the sender (6 and 7) and the sender will back down.

In both cases (ECN and BECN) the packet is marked but in case of BECN the round trip delay (3 to 7) for the source to react to congestion is avoided.

Chapter 5

Validation, Terminology used and Parameter selection

We selected Network Simulator for our congestion control study as it already has RED and ECN implemented and validated. BECN, which is a deviation of ECN, will be easier to implement. Beside's Different versions of TCP i.e. Tahoe, Reno, New Reno, and Sack are also available. This chapter first focuses on validation of BECN in NS and then describes different terminologies used in our test cases. RED parameter selection for our test cases, which plays an important role in congestion control, is described in the last section of this chapter.

5.1 Validation of BECN

The algorithm for the implementation of BECN has been described in previous chapter. For all the flows that are BECN capable the source-host will set the ECT (ECN capable transport) bit and router will mark or drop these packets in case of congestion and send an ICMP Source Quench (ISQ) back to the source. As a router validation for a BECN flow we verified that:

Total number of packet marked+dropped = Number of ISQ's generated.

Figure 5.1: Sequence versus time graph for BECN

Figure 5.1 shows Sequence Number versus time plot. As an be seen from the figure whenever a packet is marked or dropped, an ISQ is generated. According to BECN algorithm, whenever end-system receives an ISQ, its TCP would react by reducing its rate of injecting packets into the network. Figure 5.2 shows the congestion window of a BECN flow. From Figure 5.1 and 5.2, it can be seen that whenever an ISQ is sent, *cwnd* is reduced at the end hosts.

5.2 Terminology Used

This section presents the terminology used in subsequent sections and test cases.

Figure 5.2: Congestion window corresponding to Figure 5.1 BECN flow

- **Congesting flows:** These are TCP flows that are started in the background so as to create congestion in the network. FTP is used for creating congestion. The direction of congesting flows is same as of competing flows.
- **Competing flows:** These are the flows that we are interested in. They are Non-ECN, ECN, and BECN flows. They are termed competing flows because they compete for the available bandwidth in the presence of background congestion.
- **RED Region:** When the router average queue size is between the minimum threshold (min_{th}) and maximum threshold (max_{th}) of RED, we say that we are operating in the RED region.
- **Low, Medium and High Congestion:** This is defined depending on where average queue is in the RED region of min_{th} and max_{th} . For Low congestion average queue is slightly above the min_{th} , for medium congestion average queue operates midway between min_{th} and max_{th} , and for high congestion queue operates near max_{th} . Number of flows are selected depending upon the congestion level required.

- **Goodput:** We define goodput as the effective data rate as observed by the user i.e. if we transmitted 4 data packets, with two of them being retransmitted packets, the goodput is 2xpacket size/time taken to transmit four packets.

5.3 RED parameter selection

It is necessary that in the course of our experiments, the router operate in the RED region, i.e. average queue is maintained between RED parameters of min_{th} and max_{th} . If average queue goes beyond RED region, it degenerates into Drop Tail and benefits of ECN and BECN will not be experienced. Other two RED parameters are, Wq (queue weightage factor) and $Maxp$ (maximum drop probability). Parameter Wq is used for calculating average queue with each new packet arrival as:

$$\text{Average Queue} = (1-Wq) \times \text{avg. queue} + Wq \times \text{instant. queue size}$$

The value of Wq should not be too high as it will reflect the small bursts into average queue. It should not be too low so as to slow to react to traffic changes. $Maxp$ is the maximum drop probability with which a packet can be dropped in the RED region. Drop probability increases from 0 to $Maxp$ as average queue size increases from min_{th} to max_{th} . The following values of Wq and $MAXp$ has been selected for all the test cases.

$$Wq = 0.002, Maxp = 0.1$$

This value of Wq has been recommended in [1]. Tests were done in an homogeneous and heterogeneous network environment. Heterogeneous network consists of flows of either Non-ECN (TCP Reno only) and ECN (TCP Reno with ECN) or Non-ECN and BECN (TCP Reno with BECN). In homogeneous network flows were of one kind i.e. either Non-ECN or ECN or BECN only. For the heterogeneous network tests were done under Low, medium, and high level of congestion. These congestion level are in relevance

to the values of min_{th} and max_{th} . We chose the following values of min_{th} and max_{th} for this test case.

$$min_{th} = 10KB, max_{th} = 45KB.$$

Figure 5.3: Queue Variation under Low Congestion for Heterogeneous Network

Figure 5.4: Queue Variation under Medium Congestion for Heterogeneous Network

The Number of background flows were chosen so as to be in these congestion levels. Figures 5.3, 5.4, and Figure 5.5 show average and instantaneous queue sizes in these

Figure 5.5: Queue Variation under High Congestion for Heterogeneous Network

congestion levels and RED threshold parameters. For test cases involving homogeneous network, tests were done with varying number of flows and varying RTT. We don't set any levels of congestion, but set the RED threshold parameters so that average queue is always in the RED region. We chose following value of min_{th} and max_{th} for this test case.

$$min_{th} = 15KB, max_{th} = 85KB.$$

Figure 5.6: Queue Variation for 5 flows in a Homogeneous Network

Figures 5.6 and 5.7 shows the average and instantaneous queue sizes with minimum (5)

Figure 5.7: Queue Variation for 25 flows in a Homogeneous Network

and maximum number (25) of flows in a homogeneous network. It can be seen that for minimum as well as maximum number of flows average queue is always in RED region.

Chapter 6

Experiments, Results and Analysis

In this chapter we explain the simulations done in NS for comparing three different congestion control schemes i.e. Non-ECN, ECN, and BECN. Non-ECN is TCP Reno without explicit notification as in ECN or BECN. Four type of simulation experiments were conducted as a part of this research. Experiments were done with Bulk transfers as well as short transfers.

6.1 Experiment 1: Heterogeneous Network with different congestion levels

Ahmed in [38] has carried out similar experiment in Real Network with Non-ECN, ECN, and BECN capable TCP flows. We tried to validate some of the results in [38]. Figure 6.1 gives the test setup used for simulation, similar to the one used in [38]. All traffic flows from server to clients. All the links are 10Mb links except bottleneck link which is 1.5Mb from router1 to router2. Router1 is ECN or BECN capable.

Table 6.1 gives the RED parameter chosen at router1 (as specified in previous chapter). Congesting flows are started at simulation time 0sec from server to background

Figure 6.1: Network Topology for experiment 1

Congestion level	Number of congesting flows	Min_{th} (KB)	max_{th} (KB)	W_q	MAX_p	Queue limit (KB)
Low Congestion	2	10	45	0.002	0.1	60
Medium Congestion	7	10	45	0.002	0.1	60
High Congestion	14	10	45	0.002	0.1	60

Table 6.1: RED parameter settings

traffic host. After 20sec, competing flows are started from server to clients. Congesting flows are Non-ECN. Ahmed [38] and Krishnan [39] have shown that type of congesting traffic does not bias the results. Competing flows are either Non-ECN and ECN capable or Non-ECN and BECN capable. Number of congesting flows chosen for different level of congestion's shown in previous chapter are given in Table 6.1. Number of competing flows is two, one of each type under all levels of congestion. Simulation experiments are carried out under these three levels of congestion.

Test traffic is of bulk transfer type. Files of size 20MB were transferred for competing flows. Time taken to transfer this 20MB is noticed, thus giving goodput. Parameter of Goodput and Packet drops has been selected for the evaluation of Non-ECN, ECN, and BECN capable flows.

6.1.1 Results and Analysis

Figure 6.2 and 6.3 shows the graphs of goodput and packet drops respectively for Non-ECN, ECN, and BECN under low, medium, and high congestion. As seen from the graph, in terms of goodput BECN outperforms ECN and Non-ECN under all levels of congestion. ECN is also found to outperform Non-ECN as shown in some earlier studies [38][39]. They attribute ECN gain to fewer retransmissions (less packet drops figure 6.3) and timeouts as compared to Non-ECN flow. Here our focus is to find out why BECN is performing better than ECN. We analyse low congestion level case. Overall improvement

Figure 6.2: Goodput for Non-ECN,ECN,BECN under Low,medium and high congestion

of BECN over ECN can be attributed to combined effect of the following:

1. Theoretically we can approximate congestion window as:

$$cwnd = bw \times rtt$$

For low congestion, each flow should get $1.5\text{Mb}/4 = 375\text{Kb}$. Our $rtt(\text{propagation} + \text{queuing delay})$ is found to be 137ms. Therefore we have

$$cwnd = 375 \times 137 / 8000 = 6.4 \text{ packets.}$$

Figure 6.3: Packet drops for Non-ECN, ECN, BECN under Low, medium and high congestion

Figures 6.4 and 6.5 gives congestion windows (microscopic view) for ECN and BECN under low congestion. The *cwnd* for BECN is found to be hovering around calculated value. Thus due to lower time of notification of congestion i.e. 1ms for BECN (19ms for ECN), BECN fastly adapts to the network condition thereby gaining over other flows.

2. During Start-up phase of ECN, 15 packets are observed to be dropped due to exponential growth of the window and delay in notification of congestion window. This results in timeout. However for BECN due to fast congestion notification, initial window is timely adjusted towards network conditions, thus avoiding any drops.

For medium and high congestion levels BECN improvement over ECN is increased. Another factor which contributes to increase in BECN improvement under medium and high congestion is:

Figure 6.4: Congestion window for BECN for low congestion (microscopic view)

Figure 6.5: Congestion window for ECN for low congestion (microscopic view)

- Whenever $cwnd = 1$ (in packets) and that packet is marked. ECN responds to this marking by reducing $cwnd < 1$ (theoretically). Under these conditions, TCP has to wait for timer to expire before it can send any new packets (i.e. setting back $cwnd = 1$). For ECN in medium congestion it happens 4 times and under high congestion it is observed for 19 times.
- For BECN, if a packet is marked when $cwnd = 1$, an ISQ will make its $cwnd < 1$ (theoretically). When an ack comes for that packet its $cwnd$ becomes > 1 , thereby saving all the timer expiry time and contributing to its improvement over ECN. Figure 6.6 shows variation of congestion window for ECN under high congestion.

Figure 6.6: Congestion window for ECN for high congestion (microscopic view)

6.2 Experiment 2: Homogeneous network: Varying Number of flows

In this section we try to compare Non-ECN, ECN, and BECN with all the traffic of same type i.e. either all Non-ECN or all ECN or all BECN. Tests are done with varying total number of flows. Figure 6.7 shows the simulation network topology. Traffic direction, Links BW and Bottleneck link is similar to one used in previous experiment. In the

Figure 6.7: Network Topology for experiment 2

above figure n varies from 5 to 25. Only one flow is started from each server to client i.e. H_{n+1} to H_1 , H_{n+2} to H_2 H_{2n} to H_n . Different Number of flows chosen ($=n$) are 5, 10, 15, 20, 25. Table 6.2 gives RED parameter chosen as specified in previous chapter.

Initially tests were done with 5 Bulk transfer flows of either all Non-ECN, or all ECN,

min_{th} (KB)	max_{th} (KB)	W_q	$Maxp$	Queue limit (KB)
15	85	0.002	0.1	100

Table 6.2: RED settings: Experiment 2, 3, 4

or all BECN type. File size of 20MB is used for all bulk transfer flows. Goodput and packet drops were observed for each individual flow for all three types. This experiment is then repeated with 10, 15, 20, and 25 flows for each of Non-ECN, ECN, and BECN.

6.2.1 Results and Analysis

Figure 6.8 and 6.9 shows goodput (averaged over number of flows) and packet drops respectively for Non-ECN, ECN, and BECN with varying number of flows. We can make following observations:

Figure 6.8: Average Goodput for Non-ECN, ECN, BECN with varying number of flows

1. Average (Over number of flows) Goodput remains same for Non-ECN, ECN, and BECN under all levels of flows. In any number of flows, as type of flows are same thus equally competing for Bandwidth results in sharing the bandwidth. In an Heterogeneous network, for Non-ECN packet drops due to congestion can lead to

Figure 6.9: Packet drops for Non-ECN, ECN, BECN with varying number of flows

timeouts, thereby giving an opportunity for ECN and BECN to take advantage of available bandwidth.

2. In terms of packet drops, ECN and BECN outperform Non-ECN thus giving better channel utilization i.e. less packet retransmissions. However under high congestion (20-25 flows) BECN also drop (forced drops) a lot of packets. It can be attributed to the more burstiness of *cwnd* for BECN flows due to fast reaction time. As queue operates near *max* this leads to increase in forced drops.
3. Figure 6.10 shows goodput for ten individual flows for Non-ECN, ECN, and BECN. As can be seen deviation in individual goodput of Non-ECN flows is more than that of ECN and BECN. Thus in a homogeneous network ECN/BECN flows can provide more assurance for achieving fixed data rates.

Figure 6.10: Individual goodput of each flow for Non-ECN, ECN, BECN

6.3 Experiment 3: Varying RTT with fixed number of flows

In this experiment we fix number of flows to 20 and vary the propagation delay between router1 and router2. We denote by RTT the round trip propagation delay. The set up is same as in the previous experiment. Simulation experiment is carried with all the 20 flows of same type i.e. all Non-ECN or all ECN or all BECN. As in previous experiments parameters of goodput and packet drops are observed. Goodput is observed to be same as in previous experiment for 20 flows for all three types of flow. This can be attributed due to combined effect of *cwnd* and packet drops/marks:

1. When RTT is small, *cwnd* grows faster. Due to congestion, this results in large drops/marks of packets, thus affecting goodput.
2. As RTT is increased, *cwnd* rises slowly. This results in less number of dropped/marked

Figure 6.11: Packet drops for Non-ECN, ECN, BECN with variable RTT

packets. Figure 6.11 shows the average packet drops for 20 flows with variable RTT.

3. Beside's with 20 flows sharing 1.5Mb bottleneck BW, *cwnd* size of each flow is limited to a small value, thus affect of RTT variation's on *cwnd* size is limited.

From Figure 6.11, it can be seen that forced packet drops for BECN becomes same as that of ECN after RTT of 80ms. In this case reaction time to congestion for BECN is still same i.e. 2ms, but time taken to increase *cwnd* has increased significantly as compared to previous experiments (RTT 20ms). Thus RED average queue has more time to adjust to burstiness, thereby less forced drops.

6.4 Experiment 4: Short Web Transfers

This section examines the benefit of ECN, BECN for many short web tranfers under moderate congestion. Similar topology as in previous experiment is used. Two ftp flows

are started from H₄₁ and H₄₂ towards H₂₁ and H₂₂ at time 0.0. Short web transfers are done from H₂₁..H₄₀ towards H₁..H₂₀ with each client-server hosts doing five short transfers for the test period. We model 100 short web transfers by sending small number (10) of ftp packets. The starting time of short web transfer is a uniformly distributed variable in the interval (20,28) secs. Packet size for these web transfers is also an uniformly distributed random variable between (500,1000) bytes. FTP as well as web transfer flows are either Non-ECN or ECN or BECN i.e. homogeneity from previous experiment is maintained. The random packet sizes and start times are recorded and played back to be same in all the three cases.

Figure 6.12: Transfer time distribution for 100 short web transfers

6.4.1 Result and Analysis

Figure 6.12 gives the transfer time distribution (this is a plot of transfer time for each of connections in increasing order of transfer of web transfer). From this figure we note that, most of the connections have similar transfer times for all three cases. However few

connections of Non-ECN flows have higher transfer times than ECN and BECN flows and it goes to 6sec. This happens because initial round trip timer is set to 6sec in NS. If first packet of the starting flow is dropped, it invariably leads to time-out(initial *cwnd* is 1) and high transfer time due to initial TCP timer. Observing the BECN flow, it takes less time than ECN for 100 short web transfer. This is as earlier explained for Experiment 1, faster congestion notification results in better utilization of network bandwidth

6.5 Conclusions

In this we briefly summarize the conclusions made from the observations of test results of Non-ECN, ECN, and BECN.

- In a heterogeneous network (in context of Non-ECN, ECN, and BECN), BECN outperforms ECN and Non-ECN (in terms of goodput and packet drops) under all levels of congestion for bulk transfer. For short web transfers, this improvement of BECN is likely to increase due to smaller congestion window sizes as shown in [38].
- In a homogeneous network, Non-ECN, ECN, and BECN are found to give same goodput. ECN and BECN are found to have similar number of packet drops till moderate congestion, however under heavy congestion ECN drops far less number of packets than BECN. Thus in these tests ECN gives the best performance.
- Variation of RTT with same number of flows in the homogeneous network case, results in less drops for BECN. After certain RTT BECN performance is same as ECN. Goodput is not changed with RTT due to large number of flows sharing the bottleneck link.
- Short transfers flows in a homogeneous network under moderate congestion, BECN

flows gives better transaction timings than ECN and Non-ECN flows.

6.6 Future Work

In this section of the thesis, we have analyzed Backward Explicit Congestion Notification based congestion control mechanism and compared it with Explicit Congestion Notification and Non-ECN based congestion control mechanism. They all have TCPRENO form of TCP with ECN or BECN modification. The results showed a definite improvement of BECN over Non-ECN. In comparison with ECN, BECN is found to give mixed results of improvement/degradation over ECN. A future work will be to repeat this evaluation of BECN with TCP-SACK and TCP-NEWRENO based congestion control. Another interesting direction of research would be to study the effect of introducing BECN on wireless links. However, in wireless links packet loss can be due to lossy channel also, so congestion notification methods have to be carefully studied over.

BECN uses a binary feedback mechanism for congestion notification. Multi-Level BECN (MECN) uses various levels of congestion notification depending upon the severity of congestion. It can further enhance network utilization. Study of MECN and its comparative evaluation with ECN can be another direction of research.

Bibliography

- [1] Floyd S., Jacobson V., "*Random Early Detection gateways for Congestion Avoidance*" , IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.
- [2] Clark D. and Fang W., "*Explicit Allocation of Best Effort Packet Delivery Service*" , ACM Transactions on Networking, Vol. 6, No 4, August 1998, pp 362-373.
- [3] Blake S., Black D., Carlson M., Davies E., and Wang Z., Weiss W., "*An Architecture for Differentiated Services*", RFC 2475, December 1998.
- [4] Heinanen J., Baker F., Weiss W., and Wroclawski J., "*Assured Forwarding PHB Group*" , Internet RFC 2597, June 1999.
- [5] Jacobson V., Nichols K., and Poduri K., "*An Expedited Forwarding PHB*" , Internet RFC 2598, June 1999.
- [6] May M., Bolot J.C., Jean-Marie A., and Diot C., "*Simple performance Models of differentiated services schemes for the Internet*" , Proceedings of INFOCOM'99, New York, March 1999.
- [7] Goyal M., Durresli A., Jain R., and Lei C., "*Effect of Drop Precedences in Assured Forwarding*" , Internet Draft, draft-goyal-diffserv-spstdy-02.pdf, July 1999

- [8] Park W. H., Kim H., *"A study on parameter ranges of Differentiated Service mechanisms for delivering Assured Services"* Technical Report from Seoul national university.
- [9] WRED ref- http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/ios112p/gsr/wred_gs.htm
- [10] WRED parameter setting ref-[http:// www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/qcpart3/qcwred.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/qcpart3/qcwred.htm)
- [11] Seddigh N., Nandy B., Piedad P., Hadi Salim J., and Chapman A., *"An experimental study of Assured services in a Diffserv IP QoS Network"* , Proceedings of SPIE symposium on QoS issues related to The internet, Boston, November 1998.
- [12] Seddigh N., Nandy B., and Piedad P., *"Bandwidth Assurance issues for TCP flows in a Differentiated Services Network"* Proceedings of Global Internet symposium, GLOBECOM 99, Rio De Janeiro, December 99.
- [13] <http://www.netperf.org/netperf/NetperfPage.html>
- [14] Elloumi O., Snodder, and Pauwels K., *"Usefulness of three drop precedence in Assured Forwarding service"* , IETF Draft, draft-elloumi-diffserv-threestwo-00.txt, July 1999.
- [15] Yeom I. and Reddy N., *"Realizing throughput guarantees in a differentiated services network"*, Proceedings of ICMCS, June 1999.
- [16] Ibanez .J, Nichols K., *"Preliminary Simulation Evaluation of an Assured Service"* , Internet Draft, draft-ibanez-diffserv-assured-eval-00.txt, August 1998.
- [17] Thompson K, Miller G. J, Wilder R, *"Wide-Area Internet Traffic Patterns and Characteristics"* , IEEE Networks magazine, November/December 1997

- [18] Ramakrishnan k.k. and Floyd S., *"A proposal to add Explicit congestion notification(ECN) to IP"* RFC 2481, Jan 1999.
- [19] Network Simulator site <http://www-mash.cs.berkeley.edu/ns>
- [20] Bagal P., Kalyanaraman S., Packer B., *"Comparative study of RED, ECN and TCP rate control"* Technical Report, Rensselaer Polytechnic Institute, March 1999.
- [21] Nicholas K., Jacobson V., and Zhang L., *"A Two-bit Differentiated Services Architecture for the Internet"*, Internet Draft, draft-nichols-diff-svc-arch-00.txt, November 1997.
- [22] Clark D., and Wroclawski J., *"An Approach to Service Allocation in the Internet"*, Internet Draft, draft-clark-diff-svc-alloc-00.txt, July 1997
- [23] Jacobson V., *"Congestion Avoidance and Control"* , In Proceedings of SIGCOMM 88, Stanford, CA, August 1988.
- [24] Stevens W., *"TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms"* RFC 2001 Jan 1997
- [25] Allman M., Paxson V., Stevens W., *"TCP Congestion Control"*, RFC 2581, April 1999.
- [26] Baker F., *"Requirements for IP version 4 routers"*, RFC 1812, June 1995.
- [27] Floyd S., Henderson J. *"The NewReno Modification to TCP's Fast Recovery Algorithm"*, RFC 2582, April 1999.
- [28] Mathis M., Mahdavi J., Floyd S., and Romanow A., *"TCP Selective Acknowledgement options"*, RFC 2018, Oct 1996.

- [29] Ramakrishnan K.K., Floyd S., *"A proposal to add Explicit Congestion Notification to IP"*, RFC 2481, January 1999.
- [30] Salim J.H., Nandy B., Seddigh N., *"A proposal for Backward Explicit Congestion Notification for IP"*, <http://www7.nortel.com:8080/CTL/>
- [31] Clark D., Braden B., Jacobson V. et al *"Recommendations on Queue Management and Congestion Avoidance in the internet"*, RFC 2309, April 1998.
- [32] Postel J., *"Internet Control Message Protocol"*, RFC 792, September 1981.
- [33] Jain R., *"The art of Computer Systems performance analysis"*, John Wiley and sons, 1991, QA76.9.E94J32.
- [34] Prasad B., Kalayanaraman S.K., Packer B., *"Comparitive study of RED, ECN and TCP rate control"*, <http://www.packeteer.com/technology/articles.htm>
- [35] Floyd S., *"TCP and Explicit Congestion Notification"*, ACM Computer Communication Review, V. 24 N. 5, p.10-23, October 1994.
- [36] Fall K., Floyd S., *"Comparison of Tahoe, Reno and Sack TCP"*, Computer Communication Review, V.26, N.3, p. 5-21, July 1996.
- [37] UCB/LBNL/VINT Network Simulator-NS <http://www.isi.edu/nsnam/ns>
- [38] Ahmed U., *"Performance Evaluation of Explicit Congestion Notification Methods in IP Networks: ECN and BECN"*, M.Eng Thesis, Carleton University, Dec 1999.
- [39] Krishnan H., *"Analyzing Explicit Congestion Notification benefits for TCP"*, M.Sc Project report, University of California, Los Angeles, April 1998.