TCP/IP Networks with ECN over AQM

By

Rakesh M Arora

A thesis submitted to the
School of Graduate Studies and Research
In partial fulfillment of the requirement of the degree of

Master of Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Systems and Computer Engineering
Carleton University

April, 2003
©Rakesh M Arora

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

TCP/IP Networks with ECN over AQM

submitted by
Rakesh M Arora,
In partial fulfillment of the requirements
for the degree of Master of Engineering

Prof. Oliver Yang, University of Ottawa, Co-Supervisor

Prof. Thomas Kunz, Carleton University, Co-Supervisor

Chairman, Department of System and Computer Engineering
Carleton University
April, 2003

Abstract

Explicit Congestion Notification (ECN) protocol provides the mechanism for using Congestion Experienced (CE) code point in the packet header as an indication of congestion, instead of relying solely on packet drops. ECN can be used as a congestion indication policy of Active Queue Management (AQM) mechanisms e.g. Random Early Detection (RED), Dynamic RED (DRED). This thesis proposes "ECN over RED", "MECN over RED", and "ECN over DRED" protocols. We analyze and evaluate these protocols (with extensive performance characteristics: queue size, bottleneck link utilization, throughput, goodput, queue size, delay, fairness) in homogeneous and heterogeneous networks using OPNET Modeler simulation tool.

Based on the simulation results, two enhancements to the basic "ECN over RED" protocol - "Fair ECN (FECN) over RED" and "Mark First FECN (MFFECN) over RED" are proposed and analyzed. "FECN over RED" provides higher bandwidth to connections with longer Round Trip Time (RTT) when competing with lower RTT connections. This results in better fairness in a heterogeneous network. "MFFECN over RED" proposes a mark-front strategy instead of the mark-tail strategy of the standard "ECN over RED" in order to speed up the congestion notifications to the sender. These protocols have been evaluated against "ECN over RED" using the OPNET Modeler simulation tool.

Acknowledgements

I have been fortunate to have the help and support from many people during my research work. I would especially like to thank:

- Professor Oliver, W. W. Yang, my research supervisor, for providing me guidance during my research.
- Dr. Hongyi Zhang for his advice and insightful comments on my work.
- Professor Thomas Kunz for advice and understanding during my thesis work.
- Dr. J.Aweya, M.Ouellette and Dr. D. Y. Montuno of Nortel Networks for providing the OPNET TCP/IP Model.

Finally, I would like to thank my family for their unconditional support and continual encouragement.

Table Of Contents

Title Page i
Acceptance Form ii
Abstract iii
Acknowledgements iv
Table Of Contents v
List Of Figures vii
List Of Tables ix
Acronyms x
List of Notations and Symbols xi

1 Introduction 1
1.1 Overview 1
1.2 Literature Review of TCP Congestion Control 2
1.3 Motivation 8
1.4 Objectives 9
1.5 Methodologies and Approaches 10
1.6 Thesis Contributions 11
1.7 Thesis Structure 11
1.8 Publications 12

2 Network Operation, Modeling and Assumptions 13
2.1 Network Topology 13
2.1.1 Homogeneous Network Configuration 14
2.1.2 Heterogeneous Network Configuration 14
2.2 Network Operation 15
2.2.1 RED [FlJa93] 16
2.2.2 DRED [AwOu01] 17
2.2.3 Explicit Congestion Notification [Floy94, RaFl01] 19
2.3 Performance Characteristics 23
2.3.1 Queue Size 23
2.3.2 Link Utilization 23
2.3.3 Throughput 24
2.3.4 Goodput 24
2.3.5 Packet Loss Rate 24
2.3.6 Delay 24
2.3.7 Fairness 24
2.3.7.1 Jain's Fairness Index [Jain91] 25
2.3.7.2 Max-Min Fairness [PeDa00] 25
2.4 OPNET Simulation Model 25
2.4.1 Network Model 26
2.4.2 Process Model 27
2.4.2.1 TCP Source and Receiver Process Model 27
2.4.2.2 Router Process Model 29

2.5 Assumptions 30

3 ECN over AQM 31
3.1 ECN over AQM Protocols 31
3.1.1 ECN over RED 31
3.1.2 MECN over RED 32
3.1.3 ECN over DRED 32
3.2 Simulation Model 33
3.3 Simulation Scenarios 35
3.4 Performance Analysis 38
3.4.1 ECN over RED 38
3.4.2 MECN over RED 45
3.4.3 ECN over DRED 45
3.4.4 "ECN over RED" vs. "ECN over DRED" 53
3.4.5 Fairness 57
3.5 Concluding Remarks 64

4 ECN Enhancements 65
4.1 Fair ECN (FECN) 65
4.1.1 FECN over RED Algorithm 66
4.1.2 Simulation Model 68
4.1.3 Simulation Scenarios 70
4.1.4 Performance Analysis 74
4.2 Mark First Fair ECN (MFFECN) 79
4.2.1 MFFECN over RED Algorithm 80
4.2.2 Simulation Model 82
4.2.3 Simulation Scenarios 84
4.2.4 Performance Analysis 89
4.3 Concluding Remarks 94

5 Design Guidelines 95

6 Conclusions and Future Work 97
6.1 Conclusions 97
6.2 Future Work 98

7 Bibliography 99

Appendix A: TCP/IP Networks Congestion Control 102
A.1 End-to-End Congestion Control Issues 102
A.2 Traditional Congestion Control Algorithms 103
A.3 TCP Variants 107
A.4 Active Queue Management 109
A.5 Explicit Congestion Notification 110

Appendix B: Simulation results 111

List Of Figures

Figure 2-1 Homogeneous TCP/IP Network Bottleneck Configuration 13
Figure 2-2 Heterogeneous TCP/IP Network Bottleneck Configuration 14
Figure 2-3 DRED Feedback Control System 18
Figure 2-4 IP Header 21
Figure 2-5 TCP Header 22
Figure 2-6 OPNET Network Model 27
Figure 2-7 TCP/IP Source Process Model 28
Figure 2-8 IP Router Process Model 29
Figure 3-1 ECN over DRED Feedback Control System 32
Figure 3-2 RED and "ECN over RED" Performance Characteristics (100 Sources) 37
Figure 3-3 RED and "ECN over RED" Performance Characteristics (500 Sources) 40
Figure 3-4 RED and "ECN over RED" Performance Characteristics (1000 Sources) 42
Figure 3-5 DRED and "ECN over DRED" Performance Characteristics (100 Sources) 46
Figure 3-6 DRED and "ECN over DRED" Performance Characteristics (500 Sources) 48
Figure 3-7 DRED and "ECN over DRED" Performance Characteristics (1000 Sources) 50
Figure 3-8 “ECN over RED” and "ECN over DRED" Performance Characteristics (100 Sources) 53
Figure 3-9 “ECN over RED” and "ECN over DRED" Performance Characteristics (500 Sources) 54
Figure 3-10 “ECN over RED” and "ECN over DRED" Performance Characteristics (1000 Sources) 55
Figure 3-11 RED and “ECN over RED” Goodput (300 Sources) 59
Figure 3-12 RED and “ECN over RED” Goodput (500 Sources) 59
Figure 3-13 DRED and “ECN over DRED” Goodput (300 Sources) 62
Figure 3-14 DRED and “ECN over DRED” Goodput (500 Sources) 62
Figure 4-1 “ECN over RED” and "FECN over RED" Performance Characteristics (100 Sources) 71
Figure 4-2 “ECN over RED” and "FECN over RED" Performance Characteristics (500 Sources) 72
Figure 4-3 “ECN over RED” and "FECN over RED" Performance Characteristics (1000 Sources) 73
Figure 4-4 “ECN over RED” and “FECN over RED” Goodput (300 Sources) 77
Figure 4-5 “ECN over RED” and “FECN over RED” Goodput (500 Sources) 77
Figure 4-6 FECN Flow Group Queues and Router Queue 80
Figure 4-7 “ECN over RED” and "MFFECN over RED" Performance Characteristics (100 Sources) 86
Figure 4-8 “ECN over RED” and "MFFECN over RED" Performance Characteristics (500 Sources) 87
Figure 4-9 “ECN over RED” and "FECN over RED" Performance Characteristics (1000 Sources) 88
Figure 4-10 “ECN over RED” and “MFFECN over RED” Goodput (300 Sources) 92
Figure 4-11 “ECN over RED” and “MFFECN over RED” Goodput (500 Sources) 92

Figure A-1 Sliding Window Protocol 104
Figure A-2 Slow Start and Congestion Avoidance 105
Figure B-1 RED and "MECN over RED" Performance Characteristics (100 Sources) 112
Figure B-2 RED and "MECN over RED" Performance Characteristics (500 Sources) 113
Figure B-3 RED and "MECN over RED" Performance Characteristics (1000 Sources) 114

List Of Tables

Table 2-1 RED Algorithm Parameters 17
Table 2-2 DRED Algorithm Parameters 19
Table 3-1 Parameter Values for ECN over AQM algorithms 35
Table 3-2 Simulation Scenarios for ECN over AQM protocols 36
Table 3-3 Jain's Fairness Index: RED and “ECN over RED” 58
Table 3-4 Jain's Fairness Index: DRED and "ECN over DRED" 61
Table 4-1 "FECN over RED" Algorithm Parameters 68
Table 4-2 Parameter Values for "FECN over RED" algorithm 69
Table 4-3 Simulation Scenarios for "FECN over RED" 70
Table 4-4 Jain's Fairness Index: RED, “ECN over RED” and "FECN over RED" 76
Table 4-5 "MFFECN over RED" Algorithm Parameters 82
Table 4-6 Parameter Values for "MFFECN over RED" algorithm 84
Table 4-7 Simulation Scenarios for "MFFECN over RED" 85
Table 4-8 Jain's Fairness Index: RED, “ECN over RED” and "FECN over RED" 91

Acronyms

		Section of 1st appearance
ACK	Acknowledgement	1.2
AQM	Active Queue Management	1.2
ARED	Adaptive Random Early Detection	1.2
BLUE	Blue Algorithm	1.2
CE	Congestion Experienced	2.2.3
CWR	Congestion Window Reduced	2.2.3
DRED	Dynamic Random Early Detection	1.2
ECN	Explicit Congestion Notification	1.2
FACK	Forward Acknowledgement	1.2
FECN	Fair Explicit Congestion Control	1.3
FIFO	First In First Out	1.2
IETF	Internet Engineering Task Force	1.2
IP	Internet Protocol	1.2
ISP	Internet Service Provider	1.5
MECN	Marked Explicit Congestion Control	1.3
MFFECN	Mark First Fair Explicit Congestion Control	1.3
LAN	Local Area Network	1.5
MSS	Maximum Segment Size	2.2
RED	Random Early Detection	1.2
RFC	Request For Comments	6
RTT	Round Trip Time	1.2
SACK	Selective Acknowledgement	1.2
SRED	Stabilized RED	1.2
TCP	Transmission Control Protocol	1.2
UDP	User Datagram Protocol	1.1
WAN	Wide Area Network	1.5

List of Notations and Symbols

		Section of 1st Appearance
B	Buffer Size	2.2.1
P_a	RED Drop Probability	2.2.1
q_{avg}	Queue Average Size	2.2.1
$count$	Number of packets that are not dropped	2.2.1
max_p	Maximum Drop Probability	2.2.1
max_{th}	Maximum Threshold	2.2.1
min_{th}	Minimum Threshold	2.2.1
w_q	Queue Weight	2.2.1
L	No Drop Threshold	2.2.2
α	Control Gain	2.2.2
β	Filter Gain	2.2.2
Δt	Sampling Interval	2.2.2
$p_d(n)$	DRED Drop Probability	2.2.2
T	Target Queue Size	2.2.2
$\hat{e}(n)$	Filtered Error Signal	2.2.2
$q(n)$	Queue Size	2.2.2
$ssthresh$	Slow Start Threshold	3.2
N	Number of Connections	3.3
$cwnd$	Congestion Window Size	A.2

1 Introduction

1.1 Overview

Computer networking has changed enormously over the past decade, especially with the evolution of Internet. The Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite is the most popular protocol in Internet applications because it allows computers of all sizes from many different computer vendors and running totally different operating systems to communicate with each other. Initially, the best effort service model was used in the Internet for the transmission of packets. This model treats all packets equally without any discrimination or explicit delivery guarantees. The applications and protocols in this best effort service model have been considered to be flexible, adaptive and robust enough to operate under wide range of network conditions.

While the Internet traffic continues to grow, it becomes more challenging to provide fair goodput to millions of web users. The dropped packets have already consumed bandwidth on the way to the place where they are dropped, and they also cause increased incidence of TCP timeouts. In extreme cases, the situation could lead to congestion collapse [Jaco88]. The end-to-end congestion control mechanisms of TCP have been critical factors in the robustness of the Internet.

Congestion collapse occurs when an increase in the network load results in a decrease in the useful work done by the network. During the mid-1980s, the congestion collapse [Jaco88], also called “Internet meltdown” was first observed. This was largely due to TCP connections unnecessarily retransmitting packets that are in transit or had already been received by the receiver. In order to fix the congest

ion collapse problem, Jacobson [Jaco88, Jaco90] developed some congestion control mechanisms. These mechanisms operate in the end hosts to cause TCP connections to back off during congestion. Originally, TCP included window based flow control mechanisms [Stev97] as a means for the receiver to control the amount of data sent by the sender. The overflow mechanism was used to prevent the overflow of the receiver’s data buffer space available for the TCP connection. With the new congestion control mechanisms [Jaco88, Jaco90] the TCP flows are responsive to congestion signals (e.g., packet loss) from the network.

Problems with the Congestion Collapse have generally been corrected by the timer improvements and congestion control mechanisms in the modern implementations of the TCP [FlFa99]. However, with the expanding use of Internet by users and applications, it is no longer practical to rely on all end-nodes to use end-to-end congestion control for best effort traffic. Similarly, it is no longer possible to rely on all developers to incorporate end-to-end congestion control in their Internet applications.

The congestion control algorithms used by TCP results in an unfair bandwidth allocation when multiple connections share a congested gateway. TCP/IP networks have a bias against connections passing through multiple congested gateways, a bias against connections with longer roundtrip times, and a bias against bursty traffic [Floy91a]. Several techniques for measuring fairness in networks with congested gateways have been proposed [Jain91, PeBa00]. The issue of fairness among competing flows has become increasingly important with the growth of the Web. Internet users increasingly want high-bandwidth, low delay communications, rather than the leisurely transfer of a long file in the background. The growth of the best effort traffic that does not use TCP underscores this concern about fairness between best-efforts traffic in the times of congestion. For the current Internet environment, where other best effort traffic (e.g. UDP) could compete at a router queue with TCP traffic, the absence of fairness could lead to one flow starving out another flow in the time of congestion.

It has become quite clear that more control is definitely needed for protecting best effort service. The important requirement is to prevent congestion collapse, keep congestion levels low, and guarantee fairness. Congestion control and avoidance in the Internet has become one of the most important and essential technologies for data transmission over the Internet.

1.2 Literature Review of TCP Congestion Control

TCP is the dominant transport protocol used in the Internet today. As discussed above, the congestion in the TCP networks is focused on a phenomenon called congestion collapse. The end-to-end congestion control mechanisms of TCP have been critical factor in the robustness of the Internet. In the last couple of decades, various congestion

avoidance techniques have been proposed and evaluated. In this section, we review the TCP congestion control research literature.

TCP uses a window-flow-control mechanism. Since the number of connections in the network is changed randomly and the bandwidth-delay product may vary among the different network paths, it is impossible to fix the window size of each connection. Over the last couple of decades, TCP congestion control [Jaco88, Jace90, Stev97] has been used to effectively control the rates of individual connections sharing IP (Internet Protocol) network links with the TCP congestion control window. A thorough description of basic TCP algorithms is given in [Stev94, Stev97]. In TCP/IP networks, congestion control is mainly provided by end-to-end mechanisms in TCP [Jaco88, Stev97] in cooperation with queuing strategies in routers [FlJa93, Mank97]. The main idea behind the TCP congestion control is to control network load by senders' adjusting their congestion control window sizes according to the level of congestion in the network. Each sender uses its own congestion window to limit the number of unacknowledged packets that can be outstanding in the network. Upon receiving successful acknowledgements, the sender increases the size of its congestion window in order to increase the transmission rate. Sender then sends out the packets based on the new congestion window size. This continues with each successful acknowledgement to the sender. Therefore, the total transmission rate eventually exceeds the network's capacity and queues build up in the routers. Queue overflow at the routers results in packets drop and duplicate acknowledgements from the receivers. If the sender receives a given number of duplicate acknowledgements for a given packet, the sender considers that packet as lost. The sender then reduces its transmission rate by reducing its congestion window size. If the sender does not receive any acknowledgement for a specified period of time, the sender gets into a time-out period by triggering a retransmission timer. The window size is reduced to one packet and the lost packet is retransmitted. The TCP/IP network is modeled as the feedback control model in [HoMi01].

TCP congestion control methods are divided into four parts – Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery [Jaco88, Jaco90, Stev94, Stev97]. Detailed description of these congestion control techniques is summarized in

the Appendix A. The detailed behavior of TCP/IP network congestion avoidance algorithm is summarized in [MaSe97].

Various enhancements to TCP have been proposed and implemented to control network congestion while maintaining fair goodput e.g. Tahoe [Jaco88], Reno [Jaco90], New-Reno [FaFl96], Selective Acknowledgement (SACK) [FaFl96, FlMa00], Forward Acknowledgement (FACK) [MaMa96b] and Vegas [BrOM94]. In traditional TCP implementations, the go-back-N Tahoe TCP, which includes Slow Start, Congestion Avoidance, and Fast Retransmit, is implemented in the TCP sender model. The lost packets will be transmitted after the retransmission timer expires. In Reno TCP, not only these three algorithms are kept, the Fast Recovery algorithm is also used after duplicated acknowledgements (ACKs) (usually 3 ACKs) are received by the TCP sender. New-Reno TCP avoids some of the problems in the Reno TCP. When multiple packets are lost from one window, the TCP sender will remain in Fast Recovery without a retransmission timeout, and the lost packets are retransmitted one by one in each round trip time. The Fast Recovery ends with the acknowledgement of all the pending packets. In SACK TCP, instead of using the go-back-N algorithm, the selective acknowledgement and selective repeat algorithms are used. That means only the missing packets will be retransmitted. This helps TCP sender to detect network congestion quickly and avoid unnecessary retransmissions. FACK TCP, like New Reno and SACK TCP, can solve the problems caused by multiple segments loss in one window by using an additional state variable. Vegas TCP has a different window open strategy where sender increases its window size linearly when there is no congestion in the network. The performance of various TCP variants using simulation has been compared by Floyd et al. [FaFl96]. This study shows the benefit of adding SACK to TCP. In particular, it shows that TCP implementations are constrained to either retransmit at most one dropped packet per round-trip time, or to retransmit packets that might have already been successfully delivered.

Router queue management plays the key role in the congestion control and avoidance. There are several queue management algorithms implemented in current networks: Drop Tail, Drop Head, and Early Random Drop [Stev97]. Drop-Tail scheme is used by most of the available routers. The arriving packets are stored in the First-in First-

out (FIFO) queue for the particular output port. These packets are removed from the queue after transmission. Each queue has a limit on the number of packets that it can store. If a packet arrives and the queue is full then the packet is dropped.

Drop-Tail Queuing has the following problems with TCP traffic:

- It tends to drop the packets from each connection using the queue. This causes "global synchronization"[1] of connections' leading to under-utilization [Morr99a].
- The packets dropped by the drop-tail mechanism are not independent of the connections. TCP's ACK feedback mechanism causes repeating patterns or "phase effects" [FlJa92] in packet arrivals, which can cause different connections to experience different loss rate.
- TCP sources reduce their transmission only after losing packets. Therefore, even with techniques such as congestion avoidance, slow start, fast retransmit and fast recovery mechanism, the performance of TCP congestion control algorithm over drop-tail networks is inadequate in a heavily loaded network.

A variant of drop-tail called Random-Drop [Mank97] discards a randomly selected packet from the router queue instead of the packet that arrives when the queue is full. With this technique the arriving packet is queued. This approach helps eliminate phase effects but does not avoid global synchronization [FlJa93].

Sally Floyd and Van Jacobson introduce an Active Queue Management (AQM) technique called Random Early Detection (RED), for congestion avoidance in packet-switched network in [FlJa93]. Their simulations show that the RED algorithm solves two of the major problems of drop-tail and random drop gateways: bias against bursty traffic and global synchronization. The basic idea behind an AQM is to convey the congestion notification early enough to the senders, so that senders are able to reduce the transmission rates before the queue overflows and any sustained packet loss occurs. RED is recommended by the Internet Engineering Task Force (IETF) as an AQM algorithm for deployment in IP routers.

[1] Global synchronization manifests when multiple TCP hosts reduce their transmission rate in response to congestion notifications, then increasing their rate again when the congestion is reduced.

While RED queues certainly perform better than Drop-Tail queues [FlJa93], few shortcomings have been reported in recent years. With RED algorithm the resulting average queue length is quite sensitive to the level of congestion and to the RED parameter settings, and is therefore not predictable in advance [FeKa97]. S. Floyd et al. [FlGu01] propose Adaptive RED, a revised version of algorithms proposed by Feng et al. [FeKa97], to solve the problem with the minimal changes to the overall RED algorithm. A number of modifications and alternatives to RED algorithm have been proposed to overcome the possible shortcomings of RED e.g. Dynamic RED (DRED) [AwOu01], BLUE [FeKa99b], Stabilized RED (SRED) [OtLa99].

Feng et al. [FeKa97, FeKa99a] found that one of the inherent weaknesses of RED is that the effectiveness of RED depends, to a large extant, on the appropriate parameterization of the RED algorithm. For example, as the number of connections becomes large, the impact of individual congestion notification decreases. Without modifying the parameters of the RED algorithm to adopt the traffic characteristics, the RED gateway degrades into a simple Drop-Tail Gateway. On the other hand, as the number of connections becomes small, the impact of congestion notification increases. In this case, without modifying the parameters of the RED algorithm, under utilization can occur as too many sources back-off their sending rates in response to the congestion notifications. Aweya et al. [AwOu01b] propose a Load Adaptive mechanism to enhance the effectiveness of RED schemes by dynamically changing the parameters as the number of connections change. The proposed technique adjusts to the number of connections by inspecting the packet loss rate and by adjusting the thresholds to keep the loss rate at a specified value that would not cause too many timeouts.

Various variants of RED have been proposed to overcome the inherent weakness of RED and to provide good performance under different network scenarios. Several algorithms, like SRED [OtLa99] and BLUE [FeKa99], discard packets with a load-dependent probability whenever the queue buffer in the router appears to be congested. Aweya et al. [AwOu01] propose DRED, active queue management technique, which uses a feedback control approach to randomly discard packets. Zhu et al. [ZhYa01] compare the performance of DRED with RED, BLUE, and SRED. They show that DRED can stabilize the queue size very well while keeping the link utilization high helps

in controlling the packet loss rate. S. Floyd et al. [FlGu01] recognize that with RED the resulting average queue length is quite sensitive to the level of congestion and to the RED parameter settings, and is therefore not predictable in advance. They have proposed Adaptive RED (ARED) [FlGu01], which can be implemented as a simple extension within RED routers. Based on simulations, the report shows that ARED removes the sensitivity to parameters that effect RED's performance and can reliably achieve a specified target queue length in a wide variety of traffic scenarios.

One drawback of drop-tail, random-drop and AQM routers is that they must drop packets in order to signal congestion. Unfortunately these packets have already consumed bandwidth on the way to their dropped place, and they also increase TCP timeout events. An alternative strategy is to explicitly notify senders of congestion, either with a flag in the packet header, or with a special packet. Floyd [Floy94] proposed the Explicit Congestion Notification (ECN) mechanisms for TCP and evaluates the performance of ECN in TCP/IP networks using simulations. It uses RED gateways modified to set an ECN bit in the IP packet header as an indication of congestion. The report shows that ECN reduces time-outs for interactive flows, and thus provides low-delay service.

Uvaiz and Jamal [SaAh00] have shown, using experiments, that "ECN over RED" protocol improves both bulk and transactional TCP traffic over Reno TCP with one router. In the experimental test-bed, they use few homogeneous flows with one congested router. They show that there is hardly any retransmission with ECN. The main advantage of ECN is that less time is spent recovering from congestion (whereas non-ECN spends time retransmitting), and timeouts are avoided altogether. They also report that as the transactional data size increases, ECN's advantage diminishes because the probability of recovering from a Fast Retransmit increases for non-ECN.

Studies have shown that the TCP ability to share a bottleneck fairly and efficiently decreases as the number of flows increase [Jaco88]. Several researchers have studied the biased effect in TCP connections with a longer round trip time (RTT). Floyd and Jacobson [FlJa92] concluded that the RTT bias in TCP/IP networks was resulted from the TCP window increase algorithms and not related to the gateway dropping

policy. They briefly discuss how changes to the window increase could eliminate this bias.

Floyd [Floy91a] discussed the bias in TCP/IP networks against connections with multiple congested routers and connections with longer RTT, and the bias of the Tail-Drop and Random-Drop routers against bursty traffic. Using Simulations and a heuristic analysis, Floyd shows that in a network with Tail-Drop routers a longer connection with multiple congested routers can receive unacceptably low throughput. Low throughput for longer connections with multiple congested gateways is not unavoidable. However in a network with RED gateways, the Reduce-by-Half window-decrease algorithm, and the Constant-Rate window-increase algorithm [Floy91a], a connection with a long roundtrip time and multiple congested gateways can receive an acceptable level of throughput. Floyd also points out that using different measures of fairness has quite different implications. However, there is still no generally agreed upon definition of fairness in a packet network.

1.3 Motivation

Congestion control and avoidance has become one of the most important and essential technologies for the transmission over the Internet. Since the introduction of ECN, some researchers have done investigations about the behavior and performance of "ECN over RED" using simulation. These investigations mainly focus on the packet loss in a homogeneous network with a small and/or medium number of competing flows. With only these results, it is very hard to know the advantages and disadvantages of ECN algorithm. Therefore, we would like to analyze ECN mechanism with extensive performance characteristics (queue size, bottleneck link utilization, throughput, goodput, delay, fairness) in homogeneous and heterogeneous networks using OPNET Modeler simulation tool.

Standard "ECN over RED" recommends marking of incoming packets probabilistically when the average queue size is between max_{th} and min_{th}. If the average queue size exceeds max_{th}, all incoming packets are dropped. As a variant, we propose and study "Marked ECN (MECN) over RED" for performance improvements in the

TCP/IP networks. "MECN over RED" would mark all the packets if the queue size exceeds max_{th} until the router queue overflows.

Various variants of RED like Dynamic RED (DRED), Stabilized RED (SRED), Blue etc. have been proposed to overcome different shortcoming of RED. However ECN mechanism over these techniques have not been proposed and analyzed. This thesis proposes "ECN over DRED", and analyzes the behavior and performance using OPNET Modeler simulation tool.

TCP/IP networks are biased against connections with a longer RTT. Based on the "ECN over RED" simulation results, we would like to design a variant of ECN to improve the TCP fairness. We propose "Fair ECN (FECN) over RED", and analyze its behavior and performance using OPNET Modeler simulation tool. Standard ECN protocol use mark-tail strategy[2] for congestion control. We would like to design a variant of ECN to use the mark-front strategy, instead of mark-tail, to speed-up the congestion notifications to the sources. We propose "Mark First FECN (MFFECN) over RED", and analyze its behavior and performance using OPNET Modeler simulation tool.

1.4 Objectives

In general, we are interested in the analysis of the behavior and performance of ECN protocols over various popular AQM techniques (RED, DRED) in TCP/IP Networks. Specifically, we would:

- Study and analyze ECN protocol over AQM techniques such as RED, DRED in homogeneous and heterogeneous network configurations.
- Analyze the advantages and disadvantages of ECN by computing and comparing the performance of TCP/IP networks with and without the ECN protocol.
- Analyze the bias against connections with longer RTT in TCP/IP networks using ECN.
- Propose and analyze the enhancements to the existing ECN protocol to improve the performance of the TCP/IP networks.

[2] With a mark-tail strategy, the packet that just arrived will be marked, but the packets already in the buffer will be sent without being marked for congestion.

1.5 Methodologies and Approaches

In order to study ECN over AQM techniques we would like to design the bottleneck network configuration for the experiment first. This configuration would represent the interconnections of Local Area Networks (LANs) to Wide Ares Networks (WANs), or dial-up access users through WANs as in the case of Internet Service Provider (ISP) networks. The next step is to design the ECN protocol for RED and DRED, followed by the analysis of the performance characteristics that are useful for the evaluation of these protocols. We would like to use OPNET (Optimized Network Engineering Tool) Modeler [OPNET] to implement the bottleneck TCP/IP network. ECN, MECN, and other variants over RED and DRED are implemented in OPNET modeler. These protocols are implemented with various configurable parameters (e.g. number of connections, RTT). Depending upon the simulation scenario, these parameters are set at the simulation run-time. OPNET Modeler debug and animation tools were used during the experiments to ensure that these implementations are conformant to our design specifications.

To achieve the objectives listed in previous section, we conduct the following steps in our study:

1. Running the simulation and collecting the data on key performance metrics (utilization, throughput, goodput, queue size, packet loss rate, queuing delay, fairness) to evaluate the effect of ECN over AQM techniques (RED & DRED)
2. Evaluating the performance of Marked ECN (MECN) over RED by running the simulation and collecting the performance metrics.
3. Based on the results from step 1 and 2, more simulations are run to compare the performance of the ECN enhancements with the basic ECN.

For each network scenario, simulation is run for these scenarios for 100 seconds to gather the key performance data. Performance metrics are collected during simulation at a sampling interval of 10msec. Some of the metrics (e.g. Jain's Fairness Index) are calculated at the end of the simulation. For these calculations the data collected during the first 20 seconds is discarded to reduce startup and stabilization effects.

OPNET Analysis tool is used to analyze the performance data collected during simulation. This tool allows capturing of the data collected in vector and scalar files in

graphs for evaluation. These graphs in the performance analysis sections show the performance date collected every 10msec against the simulation time for different protocols.

1.6 Thesis Contributions

The contributions of this thesis are:

- Design and implementation of an OPNET simulation model for ECN algorithm over different AQM techniques (RED, DRED)
- Analysis of the behavior and performance of "ECN over RED" and "ECN over DRED" with different traffic loads (100, 500 and 1000 flows)
- Design and implementation of a simulation model for a variant of standard “ECN over RED” – “MECN over RED”, and the analysis of its behavior and performance.
- Proposal of “FECN (Fair ECN) over RED” algorithm.
- Design and implementation of a simulation model for "FECN over RED", and analysis of its behavior and performance.
- Proposal of “MFFECN (Mark First FECN) over RED” algorithm.
- Design and implementation of a simulation model for "MFFECN over RED", and analysis of its behavior and performance.

1.7 Thesis Structure

The reminder of this thesis is organized as follows:

- Chapter 2 presents Network Operation, Modeling and Assumptions. This chapter lays down the framework for our study of ECN over AQM techniques in TCP/IP Networks by describing the TCP/IP network model, experiment methodology, assumptions, performance characteristics of interest, the choice of network simulation tools, and the approaches for data analysis.
- Chapter 3 describes the “ECN over RED”, “MECN over RED” and “ECN over DRED” protocols, and analysis the behavior and performance of these protocols using the OPNET Modeler.

- Chapter 4 presents, based on the simulation results from the last chapter, two enhancements to the "ECN over RED", "FECN over RED" and "MFFECN over RED". Behavior and performance of these enhancements is analyzed using the OPNET Modeler.
- Chapter 5 provides the design guidelines for TCP/IP Networks with ECN protocols
- Chapter 6 concludes the study and makes suggestions of topics for future study.

1.8 Publications

Part of the research work has been reported in the following papers:

1. Rakesh M Arora, Oliver Yang, Hongyi Zhang. "TCP Networks with ECN over AQM". OPNETWORK 2002, August 2002, Washington DC, USA.

2 Network Operation, Modeling and Assumptions

This chapter lays down the framework for our study of ECN over AQM techniques in TCP/IP Networks by describing the Network Model, Assumptions, performance characteristics of interest, the choice of network simulation tools, and the approaches for data analysis.

2.1 Network Topology

The study uses two different network topology configurations – homogeneous network configuration and heterogeneous network configuration. These network configurations are very commonly used in the TCP/IP network research and simulation. They represent bottleneck network configuration with two routers and a number of TCP connections. Basically, this configuration can represent the interconnections of LANs to WANs or dial-up access users through WANs as in the case of ISP networks.

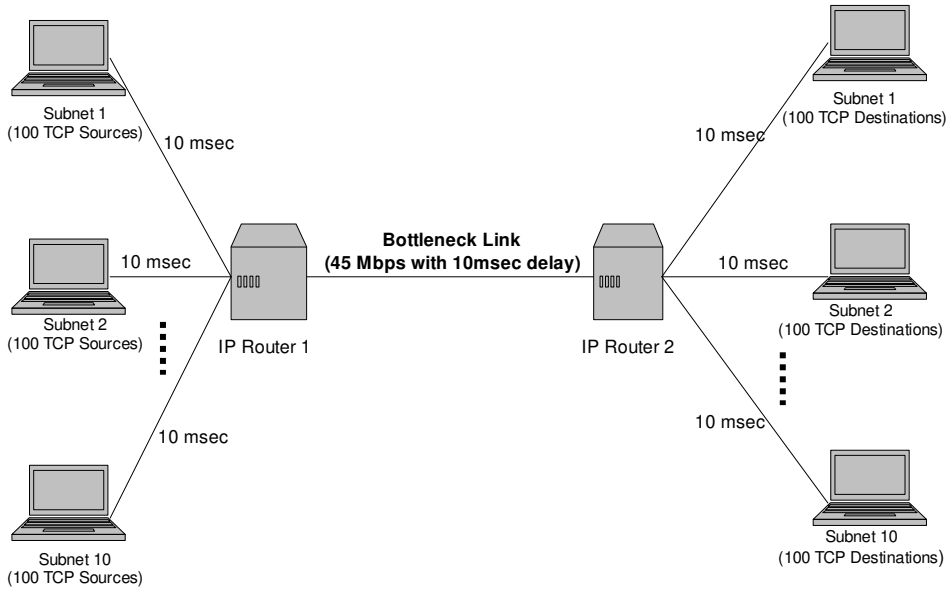


Figure 2-1 Homogeneous TCP/IP Network Bottleneck Configuration

2.1.1 Homogeneous Network Configuration

Figure 2-1 shows a network configuration with a total of 1000 TCP sources grouped into 10 subnets with 100 sources per subnet. In the homogeneous network configuration, the interconnection between the router is a T3 (45 Mbps) link and the rest of the links have equal data rate so that there is no bottleneck. All the links have a propagation delay of 10msec and the round-trip-time (RTT) is 60msec. We use this model to run the simulation with different number of sources and compare the performance of ECN algorithms over different AQM techniques.

2.1.2 Heterogeneous Network Configuration

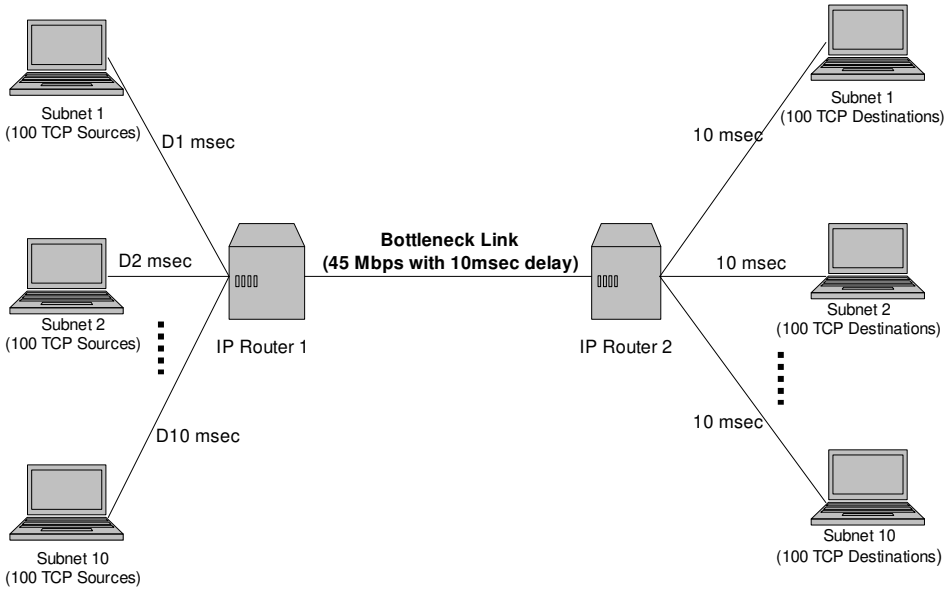


Figure 2-2 Heterogeneous TCP/IP Network Bottleneck Configuration

Figure 2-2 shows the heterogeneous network configuration. The heterogeneous network means the simulation environment consists of TCP connections with dissimilar RTTs. There are total of 1000 TCP sources grouped into 10 subnets with 100 sources per subnet. The number of connections in the network is dynamically varied during the simulations to simulate a network with different RTTs connections, and levels of traffic load. The interconnection between the router is a T3 (45 Mbps) link and the rest of the

links have equal data rate so that there is no bottleneck. The propagation delay of links from source to IP Router 1 is configurable to simulate the heterogeneous network configurations. Rests of the links have the propagation delay of 10 msec. This configuration is mainly used for analyzing the fairness in TCP/IP networks (for various protocols), when multiple connections with different RTTs are competing for the network resources.

2.2 Network Operation

TCP sources in this network configuration are based on TCP-Reno implementation, that is, they use Slow Start, Congestion Avoidance, Fast-Retransmit and Fast-Recovery mechanisms. Once the loss packet is retransmitted successfully, the TCP source will leave Fast-Recovery and get into Congestion Avoidance state with new Slow Start Threshold (*ssthresh*) and new Congestion window (*cwdn*). The maximum segment size (MSS) for TCP is set to 536 bytes, so, with the TCP header, the total TCP packet size is 576 bytes. Experiments involve scenarios with different number of TCP sources. TCP connections from these sources are modeled as greedy ftp connections; this is they always have data to send as long as their congestion windows permit. The ACK-every-segment strategy is used at the TCP destinations. The receiver's advertised window size is set sufficiently large so that TCP connections are not constrained at the destinations. TCP timer granularity 'tick' is set to 500 msec and the minimum retransmission timeout is two ticks. Based on different TCP/IP networks, the initial retransmission timeout can be 1 sec or 3 sec.

Routers in this configuration implement smarter congestion control mechanisms, AQM (RED, DRED) and ECN. These algorithms are described below in detail. These mechanisms detect incipient congestion and implicitly signals the over subscribing flows to slow down. Connection between two routers is a T3 (45Mbps) and represents the bottleneck link. Rest of the links have equal or higher data rate so that there is no bottleneck at these links. Study involves scenarios with different propagation delays for the links in the network. Buffer size of these routers is approximately equal to one delay-bandwidth product as per the industry consensus to achieve reasonable response time without the significant loss in link utilization or high packet drops.

2.2.1 RED [FlJa93]

Routers with RED Algorithm detect congestion by measuring the traffic load and inform the sources to adjust their transmission rate. It eliminates drop-tail's global synchronization and phase effects, and treats transient bursts more fairly.

RED maintains a single FIFO queue per output port and measures the traffic load by calculating average queue size q_{avg} using an exponentially weighted moving average filter as:

$$q_{avg} = (1 - w_q) * q_{avg} + w_q * q$$

where w_q is queue weight.

Network Administrator must set three RED parameters:

- min_{th}, the minimum acceptable queue length
- max_{th}, the maximum acceptable queue length
- max_p, the maximum dropping probability

Following conditions define the RED algorithm upon arrival of a packet:

- If $q_{avg} < min_{th}$, the router always queues the packet
- If $q_{avg} \geq max_{th}$, the router always drops the packet
- If $min_{th} \leq q_{avg} < max_{th}$, the router drops the packet with probability proportional to the average queue length.

The drop probability P_a is computed as:

$$P_a = \frac{P_b}{1 - count * P_b}$$

$$where\ P_b = max_p * \frac{q_{avg} - min_{th}}{max_{th} - min_{th}}$$

and *count* is a variable that keeps track of the number of packets that have been forwarded since the last drop.

RED algorithm distributes the packet drop among connections in proportion to their bandwidths as the average queue length governs the drop probability. RED averaging allows the buffering of transient bursts, as long as they are shorter than the

queue averaging interval. RED avoids global synchronization because it starts dropping packets with low probability as soon as the average queue length exceeds min_{th} to spread the packet drop over a period of time among various flows.

Table 2-1 RED Algorithm Parameters

Parameter Name	Parameter Symbol	Parameter Function	Recommended Value
Buffer Size	B	Buffer Size in router	1 or 2 times bandwidth-delay product
Maximum Threshold	max_{th}	Queue size control threshold	0.6 B
Minimum Threshold	min_{th}	Queue size control threshold	0.2 B
Maximum Drop Probability	max_p	The maximum drop probability when $min_{th} \le q_{avg} < max_{th}$,	0.1
Queue Weight	w_q	Filter Gain	0.002

The parameters used in the RED algorithm and their recommended values are shown in Table 2-1. We use the parameter values recommended by the original author [FlJa93] of the RED algorithm. Since the parameter analysis of RED algorithm has been done in many papers [FlJa93, Floy97, FiBo00], we skip the parameter sensitive analysis in this thesis. One of the RED's main weaknesses is that the average queue size is quite sensitive to the level of congestion and parameter setting. Once the average queue size exceeds the maximum threshold, the performance of RED degrades. The other major weakness is that the throughput with RED is also sensitive to traffic load and RED parameter setting.

2.2.2 DRED [AwOu01]

DRED algorithm uses a simple feedback control approach to stabilize the queue size in a router in the Internet or an Intranet. Stabilized queue size with different traffic loads is a major advantage of DRED over RED. Figure 2-3 shows the block diagram of the closed-loop feedback control system used by DRED.

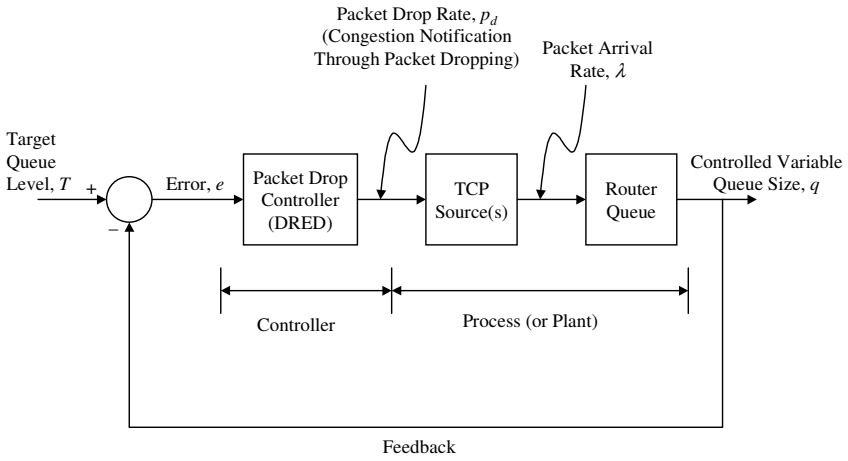


Figure 2-3 DRED Feedback Control System

The objective of DRED algorithm is to stabilize the actual queue size *q(n)* to the target queue size *T*, a threshold value independent of the network traffic load. The actual queue size is sampled every Δt units of time and fed back to produce an error signal *e(n) = q(n) - T*. This error signal is then used in the DRED controller to adapt the drop probability p_d, so that *e(n)* can be kept as small as possible. The basic strategy of the DRED algorithm is to use a discrete-time first-order low-pass filter with a filter gain β to get a filtered error signal $\hat{e}$:

$$\hat{e}(n) = (1-\beta)\hat{e}(n-1) + \beta e(n)$$

The dropping probability is then calculated using this error signal as:

$$p_d(n) = \begin{cases} 1 & \text{if } p_d(n-1) + \alpha\dfrac{\hat{e}(n)}{B} > 1 \\ p_d(n-1) + \alpha\dfrac{\hat{e}(n)}{B} & \text{if } 0 \le p_d(n-1) + \alpha\dfrac{\hat{e}(n)}{B} \le 1 \\ 0 & \text{if } p_d(n-1) + \alpha\dfrac{\hat{e}(n)}{B} < 0 \end{cases}$$

where α is the control gain and *B* is the buffer size.

DRED algorithm can stabilize [ZhYa01] queue size very well while keeping the link utilization high and helping in controlling the packet loss rate. The benefits of a stabilized queue in a network are high resource utilization, bounded delays, more certain

buffer provisioning, and traffic-load-independent network performance in terms of traffic intensity and number of TCP connections.

Table 2-2 DRED Algorithm Parameters

Parameter Name	Parameter Symbol	Parameter Function	Recommended Value
Sampling Interval	Δt	Time interval for taking measurement and applying a computed control	10 packets transmission time or any suitable value
Control Gain	α	This controls the reaction and stability of the control system	0.00005
Filter Gain	β	This controls the reaction speed of the filter	0.002
Control Target	T	This decides the buffer utilization level and average queuing delay	$T=B/2$
Buffer Size	B	Router buffer size	B = 1 or 2 times bandwidth-delay product
No-drop Threshold	L	Keep link utilization high	$L=0.9T$

All the DRED algorithm parameters and the recommended values are listed in Table 2-2. We use the parameter values recommended by the original author [AwOu01] of the DRED algorithm.

2.2.3 Explicit Congestion Notification [Floy94, RaFl01]

Explicit Congestion Notification (ECN) protocol provides the mechanism for using Congestion Experienced (CE) code point in the packet header as an indication of congestion, instead of relying solely on packet drops. ECN can be used as a congestion indication policy of AQM mechanisms (e.g. RED, DRED).

Following are the benefits of using ECN in TCP/IP networks:

- Avoid unnecessary packet drops and therefore avoids unnecessary delay for packets.
- With ECN, the congestion notification is promptly received by the TCP source, and the connection does not remain idle, waiting for the TCP retransmit timer to expire, after a packet has been dropped.

ECN requires support from both the router as well as the end hosts, i.e. the end hosts TCP/IP stack needs to be modified. Extra bits in the TCP and IP Header are required to support the ECN protocol. Router side support for ECN can be added by modifying the current AQM (e.g. RED, DRED) implementation. For packets from the ECN capable hosts, router marks the packets instead of dropping them for congestion indications. Packets from flows that are not ECN capable will continue to be dropped by the underlying AQM technique.

End hosts TCP/IP stacks are also needed to be modified to support ECN. Following is the list of these modifications:

- TCP Handshake for ECN. The source and destination TCP have to exchange information about their desire and/or capability to use ECN. Once the agreement to use ECN is reached, the sender will thereon send the ECN capable indication (ECT bit of IP Header, Figure 2-4) in all the packets.
- The receiver host conveys the congestion notification from the router to the source host by setting the ECN-Echo flag (Figure 2-5) (in the TCP Header) in the next outgoing ACK for the flow. The receiver will continue to do this in the subsequent ACKs until it receives from the sender an indication that it (the sender) has responded to the congestion notification.
- Upon receipt of the ACK with congestion indication, the sender triggers the congestion avoidance algorithm by halving its congestion window, *cwnd*, and updating its congestion window threshold value, *ssthresh*. Once it has taken the appropriate steps, the sender sets the CWR bit (Figure 2-5) on the next outgoing packet to tell the receiver that it has reacted to the receiver's notification of congestion. Note that the sender reaction to the indication of congestion in the network is equivalent to the Fast Retransmit/Recovery algorithms in non-ECN (when there is a packet loss) capable TCP, that is, the sender halves the congestion window *cwnd* and reduces the slow start threshold *ssthresh*. Fast Retransmit/Recovery is still available for ECN capable stacks for responding to three duplicate acknowledgements.

Extra bits in the TCP and IP Header are required to support the ECN protocol. Following is the list of recommendations from IETF Network Working group [RaFl01]:

- Two bits in the IP header (Figure 2-4) (bit 6 and 7 in the Ipv4 TOS octet and Traffic class octet in Ipv6, other 6 bits of this octet are reserved for Differentiated Services) are proposed for ECN Capable Transport (ECT) and Congestion Experienced (CE) bits.

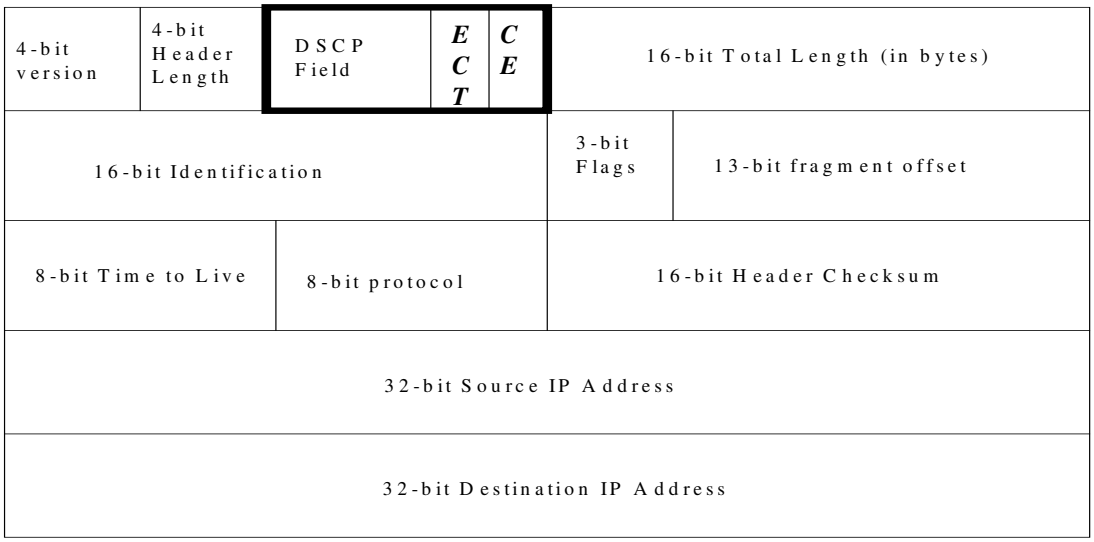


Figure 2-4 IP Header

- ECT bit is set by the sender end system if both the end systems are ECN capable.
  - ECN code points '10' – ECT (0) and '01' – ECT (1) are used by data senders to indicate that the end points of the transport protocol are ECN capable.
  - The not-ECN code point '00' indicates a packet that is not using ECN.
- Packets encountering congestion are marked by the router using the CE bit.
  - The CE code point '11' indicates (set by the router) indicate congestion to the end nodes.
- Two bits in the TCP header (Figure 2-5) to support ECN: ECN-echo flag (ECE) is bit 9 in the reserved field of TCP Header and Congestion Window Reduced (CWR) flag is bit 8 in the reserved field of TCP Header. These two bits are used for both initialization phase in which the sender and the receiver negotiate the capability and

the desire to use ECN, as well as for the subsequent actions to be taken in case there is congestion experienced in the network.

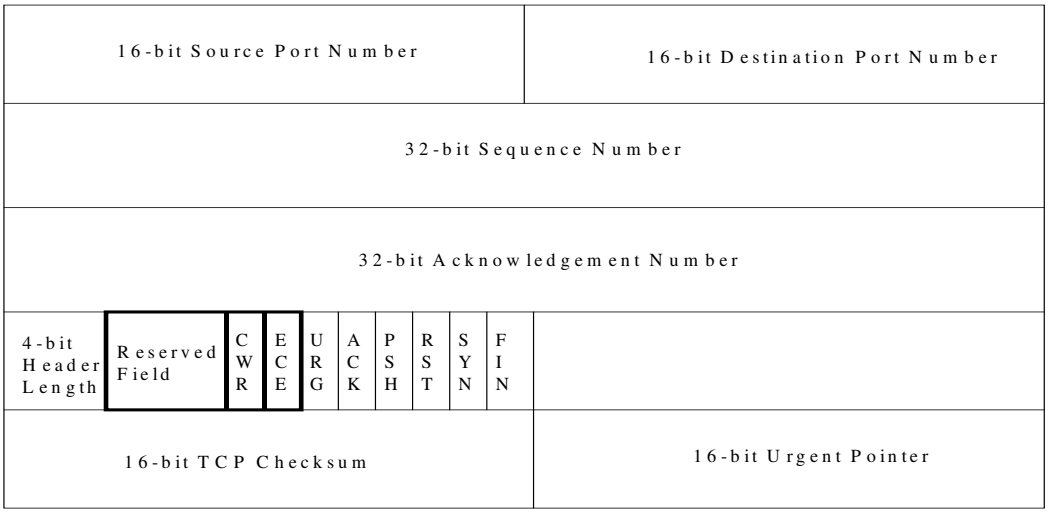


Figure 2-5 TCP Header

Following is the brief description of ECN protocol as recommended by IETF Network Working Group [RaFl01]:

- In the TCP connection setup phase both source and destination TCPs exchange information about their willingness to use ECN
  - ECN capable node sends a SYN packet with ECE and CWR flags set.
  - ECN capable node sends a SYN-ACK packet with ECE flag set but CWR flag not set.
- An ECT code point, ECT(0) or ECT(1) is set in packets by the sender to indicate that ECN is supported by the transport entities for these packets
- An ECN-capable router detects impending congestion (by using AQM protocol RED, DRED) and detects that ECT code point is set in the packet it is about to drop. Instead of dropping the packet, the router chooses to set the CE code point in the IP header and forwards the packet.
- Receiver receives the packet with the CE code point set, and sets the ECE flag in the next TCP ACK sent to the sender.

- Upon the receipt of a TCP ACK with ECE flag set, the congestion control algorithms followed by the TCP sources is same as the congestion control response to a single packet loss (halve its congestion window)
- The sender sets the CWR flag in the TCP header of the next packet sent to the receiver to acknowledge its receipt of and reaction to the ECE flag.

2.3 Performance Characteristics

The behavior and performance of the ECN and other variants over AQM techniques is evaluated by measuring and calculating various performance metrics (queue size, bottleneck link utilization, throughput, goodput, packet loss rate, delay, fairness) during the course of simulation. This section describes these performance indicators.

2.3.1 Queue Size

Queue size indicates the number of pending packets in the Router queue. Router queue size is an important performance characteristic of the TCP/IP networks. The main idea behind the active queue management to detect incipient congestion early is based on the queue size. One of the objectives of network congestion control is to stabilize the queue size, as an unstable system often leads to queue oscillations and strong synchronization among TCP flows [FlJa92]. The queue size is plotted with a 10 msec sampling interval

2.3.2 Link Utilization

Link Utilization indicates the utilization of the bottleneck link in our network model. The good utilization of the bottleneck link results in better performance for the network traffic. The objective is to study the link utilization for various techniques in congestion with other performance characteristics listed in this section. Link Utilization is calculated and plotted with a 10 msec sampling interval.

$$Link\ Utilization\ (\%) = \frac{Throughput * Packet\ Size}{Link\ Capacity} * 100$$

2.3.3 Throughput

Throughput is defined as the data rate at which a source can send packets (including retransmitted packets) to the sink. For example, assume a source sends out *m* packets in time *t*, and *n* packets ($n \prec m$) are dropped in the course of transmission, then the resulting throughput is *(m-n) * packet-size/t*.

2.3.4 Goodput

Goodput is defined as the effective data rate as observed by the user. For example, assume *m* data packets are transmitted from a source to a sink, and *n* packets ($n \prec m$) are retransmitted packets, then the resulting goodput is *(m-n)*packet-size/t*. Goodput indicates the efficiency of the network. Goodput is normally lower than throughput as it does not include the retransmitted packets.

2.3.5 Packet Loss Rate

Packet loss rate indicates the number of packets that are dropped due to router buffer overflow and for congestion notifications (in AQM techniques).

$$Packets\ Loss\ Rate = \frac{Number\ of\ \ Drop\ Packets}{Total\ Number\ of\ \ Packets} * 100$$

Packet loss results in retransmissions, which in turn results in lower goodput and efficiency. One of the objectives of ECN is to reduce the packet loss rate by marking the packets for congestion notification instead of dropping them.

2.3.6 Delay

Delay indicates the queue delay in our simulation results. It can be used to calculate the total packet delay from source to sink by including the link propagation delay. Link Propagation delay is configurable in the network model used for the simulation. Delay is an important characteristic for many applications and Internet users.

2.3.7 Fairness

This metrics indicates the fairness of the bandwidth allocation among heterogeneous flows sharing a congested gateway. Fairness has been defined in a number of different ways. However, there is still no general agreed upon definition for fairness in a computer

network. Different measurements of fairness have quite different implications [Floy91a]. In this thesis, two popular fairness measurement methods are used: Jain's fairness index [Jain91] and max-min fairness [PeDa00].

2.3.7.1 Jain's Fairness Index [Jain91]

Jain's fairness index postulates that the network is a multi-user system, and drives the metric to see how fairly each user is treated. Jain's fairness index is a function of variability of throughput across different users. For a set of user throughput ($x_1, x_2, ..., x_n$), Jain's fairness index to the set of users is defined as follows:

$$f(x_1, x_{2,}...., x_n) = (\sum_{i=1}^{n} x_i)^2 / (n * \sum_{i=1}^{n} x_i^2)$$

The fairness index always lies between 0 and 1. A value of 1 indicates that all flows get exactly the same throughput.

2.3.7.2 Max-Min Fairness [PeDa00]

Max-Min Fairness is another common fairness definition, which is also called bottleneck optimality criterion. A feasible flow rate x is defined to be max-min fair if any rate x_i cannot be increased without decreasing some x_j which is smaller than or equal to x_i.

To satisfy the max-min fairness criteria the smallest throughput rate must be as large as possible. Given this condition, the next smallest throughput must be as large as possible, and so on [Floy94]. In this thesis, graphs are used to visually analyze the max-min fairness with respect to throughput and goodput for different heterogeneous flows.

2.4 OPNET Simulation Model

This study used OPNET Modeler, version 7.0 to implement the TCP/IP network simulation model. OPNET Modeler is a comprehensive and commercial network technology development environment allowing the users to design and study communication networks, devices, protocols and applications. OPNET was originally developed at MIT, and originally introduced in 1987 as first commercial network simulator. In recent years OPNET Modeler has emerged as one of the leading simulation environment of communication systems. If offers highly graphical interfaces that

facilitates the object-oriented modeling of hierarchical network models. OPNET Modeler is based on a series of hierarchical editors (Network Editor, Node Editor, and Process Editor) that directly parallel the structure of real networks, equipments, and protocols.

Following are the main features of OPNET Modeler [OPNET]:

- Graphical Environment that models real networks, devices, protocols, and applications.
- Control over modeling details needed to support the engineering decisions.
- Build-in support to model major engineering technologies.
- Comprehensive library of standard based protocols model, with completely open source code.
- An environment designed to model proprietary protocols using Finite State Model, C/C++ and extensive libraries.
- Powerful analysis tool integrated into the GUI.
- Efficient Simulation Engine with hybrid simulation capability.

This section describes the OPNET simulation models (Network Model, Node Model and Process Model), used in this thesis for evaluating the performance of TCP/IP Networks with different AQM and ECN techniques. The TCP/IP AQM model [ZhYa01] is enhanced to include ECN protocols over RED and DRED.

2.4.1 Network Model

Figure 2-6 shows the OPNET Network Model that represents a bottleneck network configuration and is used performance evaluation of different protocols. It has two routers and a number of subnet nodes. Each source and destination subnet has 100 TCP nodes. TCP sources are based on TCP-Reno implementation, that is, they use Slow Start, Congestion Avoidance, Fast-Retransmit, Fast-Recovery mechanisms. The bottleneck link is the connection between two routers, rt-1 and rt-2. This configuration can represent the interconnections of LANs to WANs or dial-up access users through WANs as in case of ISP network. The TCP parameters are defined and initialized in TCP parameter node model.

The routers are connected through a T3 (45 Mbps) link. All other links have equal speed such that they do not create any bottleneck. The allocated buffer size of the bottleneck router (rt_1) is set to twice the bandwidth delay product (BDP) of the network. The propagation delay of the links (default: 10 ms) and RTT (default: 60ms) are configurable and can be set differently for subnets to simulate heterogeneous flows.

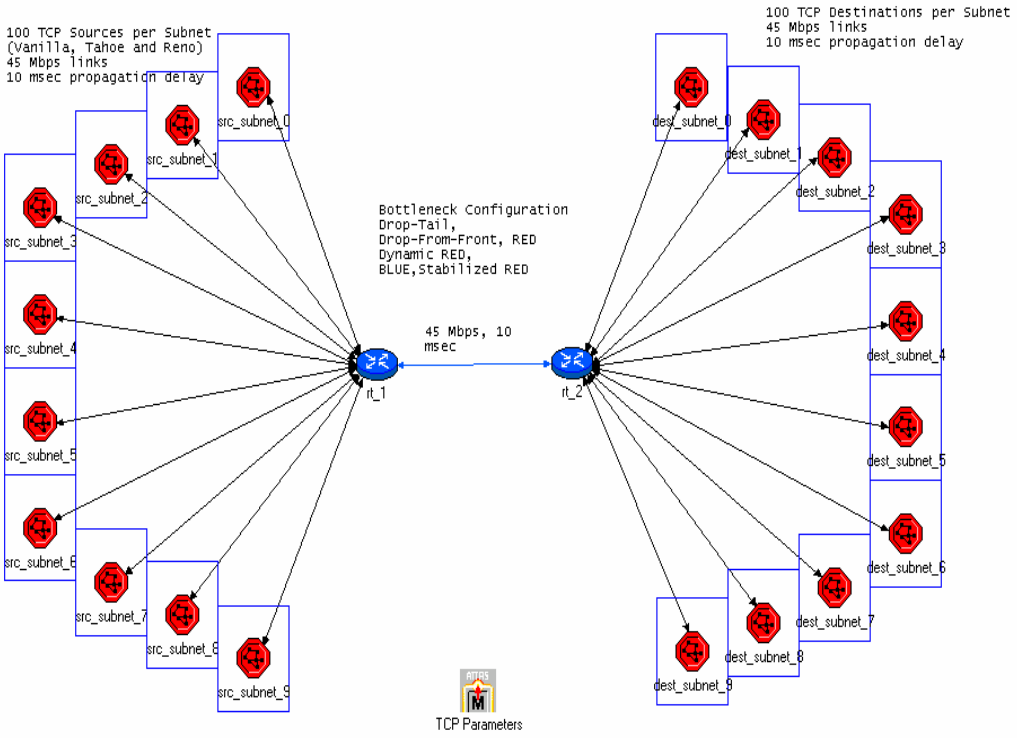


Figure 2-6 OPNET Network Model

2.4.2 Process Model

This section describes the OPNET Process Model for TCP source, receiver and router.

2.4.2.1 TCP Source and Receiver Process Model

The TCP sources are based on a TCP-Reno implementation. The Reno version uses the Fast-Retransmit and Fast-Recovery mechanisms. The TCP connections are modeled as greedy FTP connections; that is, they always have data to send as long as their connection window permits. The maximum segment size (MSS) for TCP is set to 536

bytes. The receiver's advertised window size is set sufficiently large (10000 packets for this study) so that TCP connections are not constrained at the destinations. The Ack-every segment strategy is used at the TCP receivers. The TCP time granularity "tick" is set to 500 ms and the minimum transmission timeout is set to two ticks.

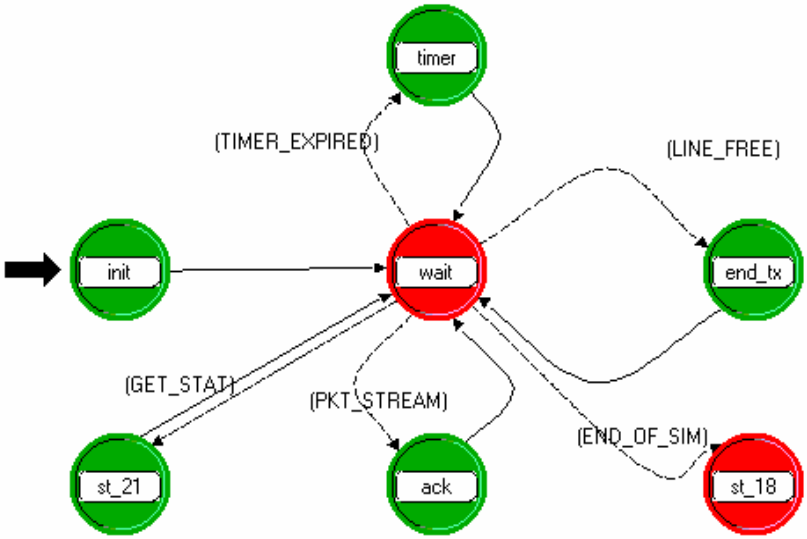


Figure 2-7 TCP/IP Source Process Model

Figure 2-7 shows the OPNET TCP/IP source Process Model. This model is used for the TCP/IP sources in the network model "src_subnet_n" nodes. The process model implements the TCP-Reno version (i.e., with fast retransmit and fast recovery). The *"init"* state is used to initialize the source parameters for the simulation. The *"end_tx"* state is used to send out the TCP packet. *"timer"* state is to calculate the timeout period. The *"ack"* state is used when acknowledgements are received from the destination. Each acknowledgement is checked for congestion notification and processed. It transits into the *"wait"* state automatically when no processing is to be performed. The *"st_21"* state is to collect statistic values at specified intervals and, the *"get_stat1"* and "*get-stat2*" are to collect statistic values at the end of a simulation.

2.4.2.2 Router Process Model

The IP Router in the Network Model implements the AQM (RED, DRED) and ECN techniques. The allocated buffer size of the bottleneck router (rt_1) is set to twice the bandwidth delay product (BDP) of the network.

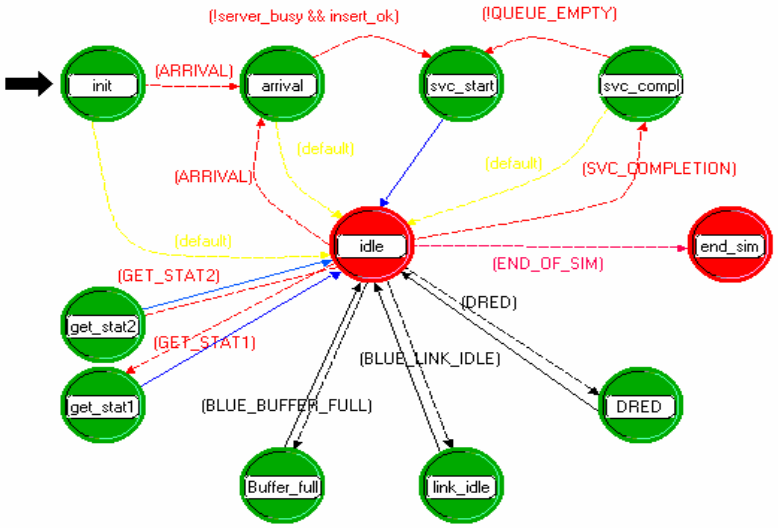


Figure 2-8 IP Router Process Model

Figure 2-8 shows the OPNET IP router Process Model. This process model is used for the router nodes rt_1 and rt_2 of the Network Model. The process model of the router, which is an extension the M/M/1 queue model, consists of five main states – *"init"* state, *"arrival"* state, *"svc_start"* state, *"svc_compl"* state and *"idle"* state. The *"init"* state is used to initialize the parameters for the simulation. In the *"arrival"* state, instead of using the drop tail method of the acb_fifo model, the AQM (RED and DRED) and ECN algorithms, described in the previous section, are implemented. The packet service process and sending-out process are implemented in the *"svc_start"* state and the *"svc_compl"* state respectively. It transits into the *"idle"* state automatically after the processing. "DRED" state is used to implement the "ECN over DRED" protocol feedback control and calculates the error signal and drop probability. The *"get_stat1"* state and the *"get_stat2"* state are used to collect statistic values at certain intervals.

2.5 Assumptions

This study has the following main assumptions for the network configuration:

- In all the network models, all sources have one to one mapping to the destinations, i.e. the traffic from each source is directed only to the corresponding destination.
- Sender always has packet to send and will send as many packets as its window allows.
- Router queue length is measured in packets and all packets have the same size.
- Receiver's advertised window size is set sufficiently large (10000 packets for this study) so that the TCP connections are not constrained at the destination.
- Receivers acknowledge every received packet and there are no delayed ACKs.
- The buffer size in IP Router 2 (rt-2 of Figure 2-6) is large enough (set to infinity for this study) so there is no packet loss at the router.

3 ECN over AQM

This chapter describes the “ECN over RED”, “MECN over RED” and “ECN over DRED” protocols for this study. Simulation scenarios are defined to evaluate the performance of these protocols. Performance results from the simulation runs are presented and analyzed.

3.1 ECN over AQM Protocols

ECN protocol provides the mechanism for using Congestion Experienced (CE) code point in the packet header as an indication of congestion, instead of relying solely on packet drops. ECN can be used as a congestion indication policy of AQM mechanisms (e.g. RED, DRED). This may improve the network performance as the unnecessary packet drops to indicate congestion are avoided. This also results in prompt congestion notifications to the TCP sources, and the connection does not remain idle, waiting for the TCP retransmit timer to expire, after the packet has been dropped. This section describes these protocols.

3.1.1 ECN over RED

"ECN over RED" provides a lightweight mechanism for routers to send a direct indication of congestion to the source. "ECN over RED" calculates the average queue size q_{avg} by using the exponentially weighted moving average filter as:

$$q_{avg} = (1 - w_q) * q_{avg} + w_q * (q - queue_ce_count)$$

where *queue_ce_count* is the number of marked packets (for congestion notifications) in the route queue.

“ECN over RED” requires the same parameters (min_{th}, max_{th}, max_p) as RED (refer Section 2.2.1). However, the following conditions define the “ECN over RED” algorithm upon arrival of the packet:

- If $q_{avg} < min_{th}$, the router always queues the packet
- If $q_{avg} \geq max_{th}$, the router always drops the packet

- If $min_{th} \leq q_{avg} < max_{th}$, the router marks the packet for congestion with probability proportional to the average queue length, and queues the packet.

The marking probability P_a is computed in the same way as for RED (refer Section 2.2.1).

3.1.2 MECN over RED

In this study, one variant of "ECN over RED", called MECN (Marked ECN) is also investigated to compare its behavior with standard ECN as recommended in [RaFl01]. MECN differs from ECN in that MECN marks packets when the average queue size exceeds max_{th} and drops packets only when the router queue overflows.

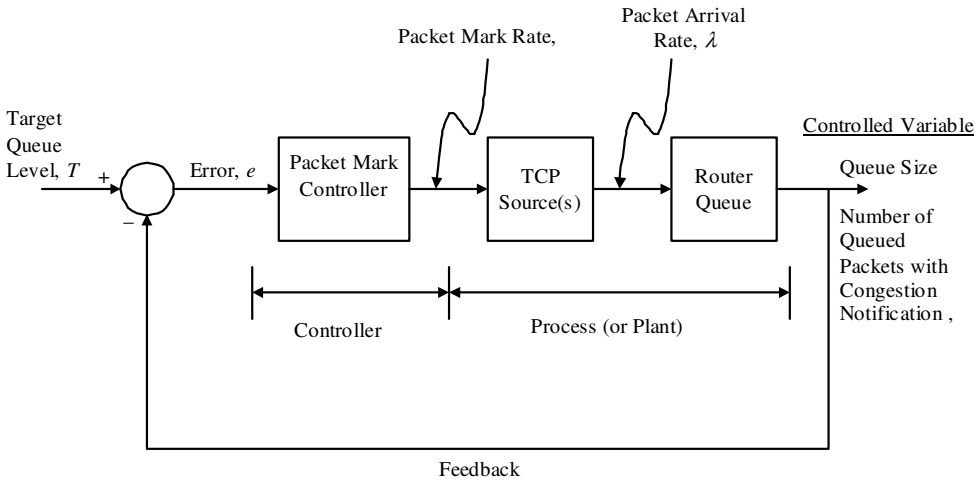


Figure 3-1 ECN over DRED Feedback Control System

3.1.3 ECN over DRED

"ECN over DRED" algorithm uses a simple feedback control approach to randomly mark packets to indicate congestion. Figure 3-1 shows the block diagram of the closed-loop feedback control system used by ECN over DRED at the router. Note that the "ECN over DRED" control system differs from DRED in the following two aspects:

- Packets are marked for congestion and queued to send the congestion notifications to TCP sources.

- Number of Queued packets with congestion notifications are used to compute the error signal, to eliminate the effect of these packets on the error signal and the congestion notification probability

The objective of "ECN over DRED" is to reduce the packet drop rate while maintaining the actual queue size *q(n)* stability (equal to the target queue size *T*, a threshold value independent of the network traffic load) of DRED. The actual queue size is sampled every Δt units of time and fed back to produce an error signal $e(n) = q(n) - c(n) - T$, where *c(n)* is the number of packets with congestion notification in the queue. Parameter *c(n)*, number of packets with congestion notification in the queue, is required for "ECN over DRED" to eliminate the effect of these packets on the error signal and the congestion notification probability. This is because the queue size includes these packets and without this parameter error signal will not correctly represents the deviation from the target queue size. This error signal is then used in the controller to adapt the congestion notification probability P_d, so that *e(n)* can be kept as small as possible. Technique to calculate the congestion notification probability is same as that for DRED algorithm (refer Section 2.2.2).

Once the queue size increases the target queue size, the incoming packets are marked for congestion (ECN flag) with the congestion notification probability. Upon the receipt of a TCP packet with congestion notification from router, the TCP receiver host sets the ECN-Echo flag in the next ACK packet to the TCP sender. Upon the receipt of a TCP ACK with ECN-Echo flag set, the congestion control algorithms followed by the TCP sources is same as the congestion control response to a single packet loss (halve its congestion window). The sender sets the CWR flag in the TCP header of the next packet sent to the receiver to acknowledge its receipt of and reaction to the ECN flag.

3.2 Simulation Model

OPNET Network Model described in Section 2.4.1 is used for the performance evaluation of the ECN over AQM protocols listed above. TCP Source, Router and Receiver process models defined in Section 2.4.2 are enhanced to implement "ECN over

RED", "MECN over RED" and DRED over RED". This section describes the changes in these models to support these protocols.

Following is the list of changes for the TCP Source Process Model:

- All the packets are sent with ECN enable bit set in the IP header block
- These protocols are used together with TCP Congestion Control mechanisms like slow-start and congestion avoidance. When an acknowledgement is not marked for congestion, the source follows the existing TCP algorithm to send data and increase the congestion window.
- Upon the receipt of ACK packet with Congestion Notification (CE bit set to 1), the source halves its congestion window and reduces the *ssthresh*. In the case of packet loss, the source follows the TCP algorithm to reduce the window and retransmit the lost packet.

IP Router process implements various AQM and ECN techniques. The congestion control technique use by the router is configurable at the simulation run-time. Following is the list of changes in the router process model to support these protocols:

- If $q_{avg} < min_{th}$ ("ECN over RED, "MECN over RED"), the router always queues the packet
- If $q_{avg} \geq max_{th}$ ("ECN over RED, "MECN over RED"), the router drops the packet for "ECN over RED". However, packets is marked for congestion and queued for "MECN over RED".
- If $min_{th} \leq q_{avg} < max_{th}$ ("ECN over RED, "MECN over RED") OR

  Queue_Size > Target Queue Size ("ECN over DRED")
  - Calculate the marking probability as per the algorithm defined for the particular congestion technique
  - Packet is marked for congestion and queued.

TCP Receiver process model checks for congestion notification in all the packets and forwards the notifications to the source by setting the ECN-Echo flag in the next ACK packet to the TCP sender. The receiver will continue to do this in the subsequent ACKs until it receives from the sender an indication that it (the sender) has responded to the congestion notification.

Table 3-1 Parameter Values for ECN over AQM algorithms

Algorithm	Parameters	Values
ECN over RED and MECN over RED	Minimum Threshold (min_{th})	118 packets
	Maximum Threshold (max_{th})	352 packets
	Maximum Drop Probability (max_p)	0.1
	Queue Weight (w_q)	0.002
	Buffer Size (B)	586 packets
ECN over DRED	Sampling Interval (Δt)	10 packet transmission time
	Control Gain (α)	0.00005
	Filter Gain (β)	0.002
	Control Target (T)	293 packets
	Buffer Size (B)	586 packets
	No-drop Threshold (L)	264 packets
	Initial Drop Probability	0.05

The parameter values of the algorithms analyzed in this chapter are shown in Table 3-1. These parameters have been described in Section 2.2. The parameter values for these algorithms have been derived using the recommended values from the original papers [FlJa93, AwOu01]. Note that the parameter analysis of RED algorithm has been done in lot of papers [Floy97, FlJa93, FiBo00]; we do not focus on the parameter sensitive analysis of these algorithms in this thesis. Since the impact of these parameters on "ECN over RED", "MECN over RED" and "ECN over DRED" is same as that on their base protocols (RED and DRED), it is reasonable to use the recommended values from the original papers.

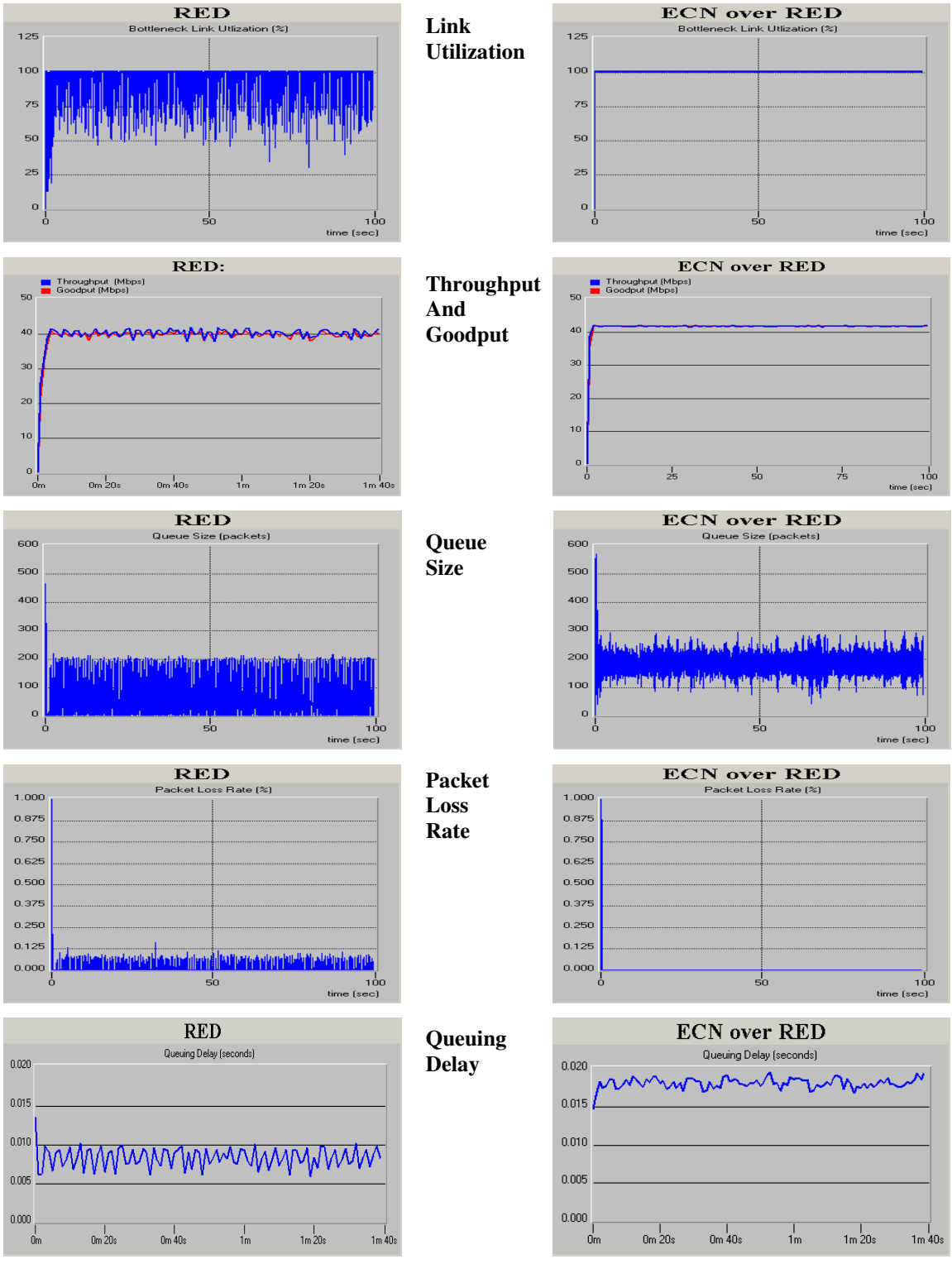
3.3 Simulation Scenarios

To evaluate the performance of "ECN over RED", "MECN over RED" and "ECN over DRED", a series of simulations are run with OPNET Modeler. We run the simulation on PIII-500 NT workstation. It takes around 20 minutes to run a simulation for 100 simulation seconds. Depending upon the simulation scenario the configurable parameters for the OPNET model (network, node and process model) described above are set at the simulation run time. Simulation is run for 100 seconds for each scenario to gather the key performance data.

Table 3-2 Simulation Scenarios for ECN over AQM protocols

	Protocol	Network Configuration	Sources	Conn-ections (N)	RTT (msec)
Scenario 1a	RED and "ECN over RED"	Homogeneous	100	100	60
Scenario 1b	RED and "ECN over RED"	Homogeneous	500	500	60
Scenario 1c	RED and "ECN over RED"	Homogeneous	1000	1000	60
Scenario 2a	RED and "MECN over RED"	Homogeneous	100	100	60
Scenario 2b	RED and "MECN over RED"	Homogeneous	500	500	60
Scenario 2c	RED "MECN over RED"	Homogeneous	500	500	60
Scenario 3a	DRED and "ECN over DRED"	Homogeneous	100	100	60
Scenario 3b	DRED and "ECN over DRED"	Homogeneous	500	500	60
Scenario 3c	DRED and "ECN over DRED"	Homogeneous	500	500	60
Scenario 4a	RED and "ECN over RED"	Heterogeneous	300	100	60
				100	100
				100	140
Scenario 4b	RED and "ECN over RED"	Heterogeneous	500	100	60
				100	80
				100	100
				100	120
				100	140
Scenario 5a	DRED and "ECN over DRED"	Heterogeneous	300	100	60
				100	100
				100	140
Scenario 5b	DRED and "ECN over DRED"	Heterogeneous	500	100	60
				100	80
				100	100
				100	120
				100	140

Simulation scenarios for evaluating the performance of various ECN over AQM protocols are listed in Table 3-2.



(a) RED **(b) ECN over RED**

Figure 3-2 RED and "ECN over RED" Performance Characteristics (100 Sources)

3.4 Performance Analysis

3.4.1 ECN over RED

Figure 3-2 shows the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for RED and “ECN over RED” algorithms for 100 sources scenario. Note that the sources are modeled as greedy FTP connections; that is, they always have data to send as long as their congestion window permits.

The “Link Utilization” graphs in Figure 3-2 show the bottleneck link utilization in our network model for RED and "ECN over RED" protocols. Bottleneck link should be optimally utilized for better network performance. For “RED” protocol, the bottleneck link is not 100% utilized. Link utilization for “ECN over RED” is better than “RED".

The "Throughput and Goodput" graphs in Figure 3-2 show the throughput and goodput performance characteristics for RED and "ECN over RED" protocols for 100 TCP sources scenario. Refer to Section 2.3 for description of these performance characteristics. Graphs indicate that throughput and goodput for "ECN over RED" is higher compared to RED. It means that the effective data rate is higher for "ECN over RED" protocol. This is because, for "ECN over RED" protocol, the router marks and forwards the packets instead of dropping the packets, resulting in higher data rate. Note that the difference between throughput and goodput represents the number of retransmit packets. For the 100 sources scenario, total number of retransmit packets is slightly higher for RED protocol compared to "ECN over RED".

The "Queue Size" graphs in Figure 3-2 indicate the router queue size performance. The queue size for “ECN over RED” is higher than the RED protocol. This is because the extra packets are marked and queued instead of being dropped. Graphs also indicate that quite size oscillates a lot for both RED and "ECN over RED" protocol. At higher loads, the oscillations in the system can push the operating queue size beyond the maximum buffer size, resulting in packet drops.

The "Packet Loss Rate" graphs in Figure 3-2 show the number of packets dropped by the router due to congestion and queue overflow. Graphs indicate that the packet loss rate is higher for RED protocol compared to "ECN over RED". As the "Queue Size" graphs indicate that there is no router queue overflow for 100 sources

scenario, the packet drop rate for "ECN over RED" protocol is 0. Packets are dropped by the RED protocol for congestion control (to indicate congestion).

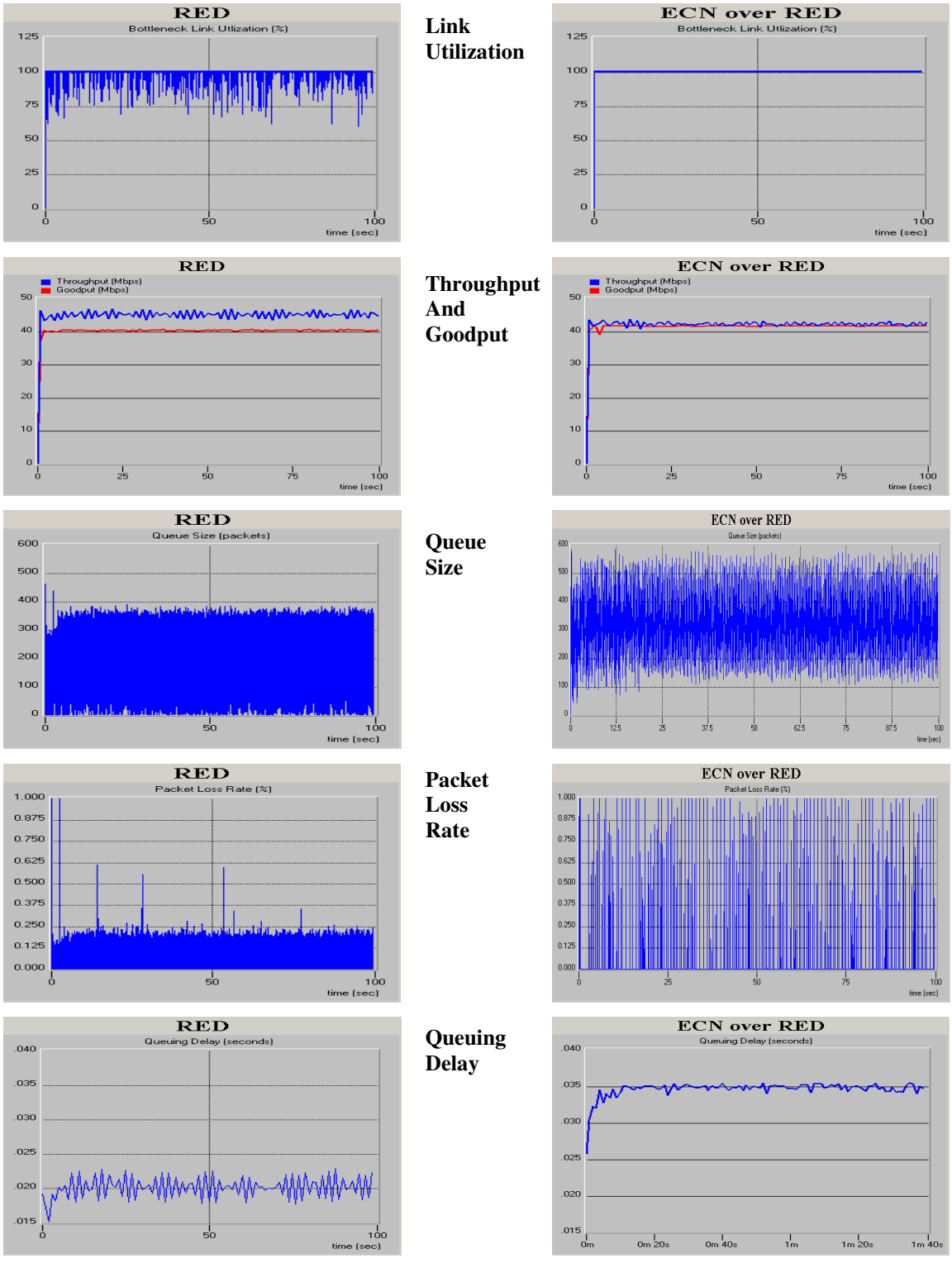
The "Queuing Delay" graphs in Figure 3-2 indicate the router queuing delay for 100 sources scenario. For “ECN over RED” protocol, the delay is higher compared to the RED protocol. This is because of the higher queue size resulting in longer time for the packets in the router queue waiting for processing.

Figure 3-3 shows the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for RED and “ECN over RED” protocol for 500 sources.

The link utilization for “ECN over RED” is better than RED. With 500 sources the bottleneck link is better utilized for RED protocol compared to the 100 sources scenario. This is because of the higher input date rate for the router and the bottleneck link.

The "Throughput and Goodput" graphs indicate that the goodput i.e. the effective data rate for "ECN over RED" protocol is higher compared to RED protocol. Graphs also indicate that the difference between throughput and goodput, i.e. the number of retransmit packets, is quite high for RED protocol compared to the 100 sources scenario (Figure 3-2). This is because of the higher input date rate to the router resulting in high number of packet drops to indicate congestion. The difference between throughput and goodput is quite low for "ECN over RED" protocol. This is a major advantage of "ECN over RED" protocol as it results in lower number of retransmissions and higher data rate.

Queue size for "ECN over RED" protocol is higher than the RED protocol. This is because of the extra packets that are marked and queued instead of dropped, due to congestion, for “ECN over RED” protocol. Queue size for both RED and "ECN over RED" is higher for 500 sources scenario compared to the 100 sources scenario. Queue size of RED depends upon the number of sources i.e. active connections. Graphs also indicate that queue size oscillates a lot for both RED and "ECN over RED" protocol. Unstable router queue size with RED protocol is well known [ZhYa01]. The queue size with “ECN over RED” is equally unstable as it still depends upon the number of connections, which is due to the design principle of RED inherited by “ECN over RED”.



(a) RED **(b) ECN over RED**

Figure 3-3 RED and "ECN over RED" Performance Characteristics (500 Sources)

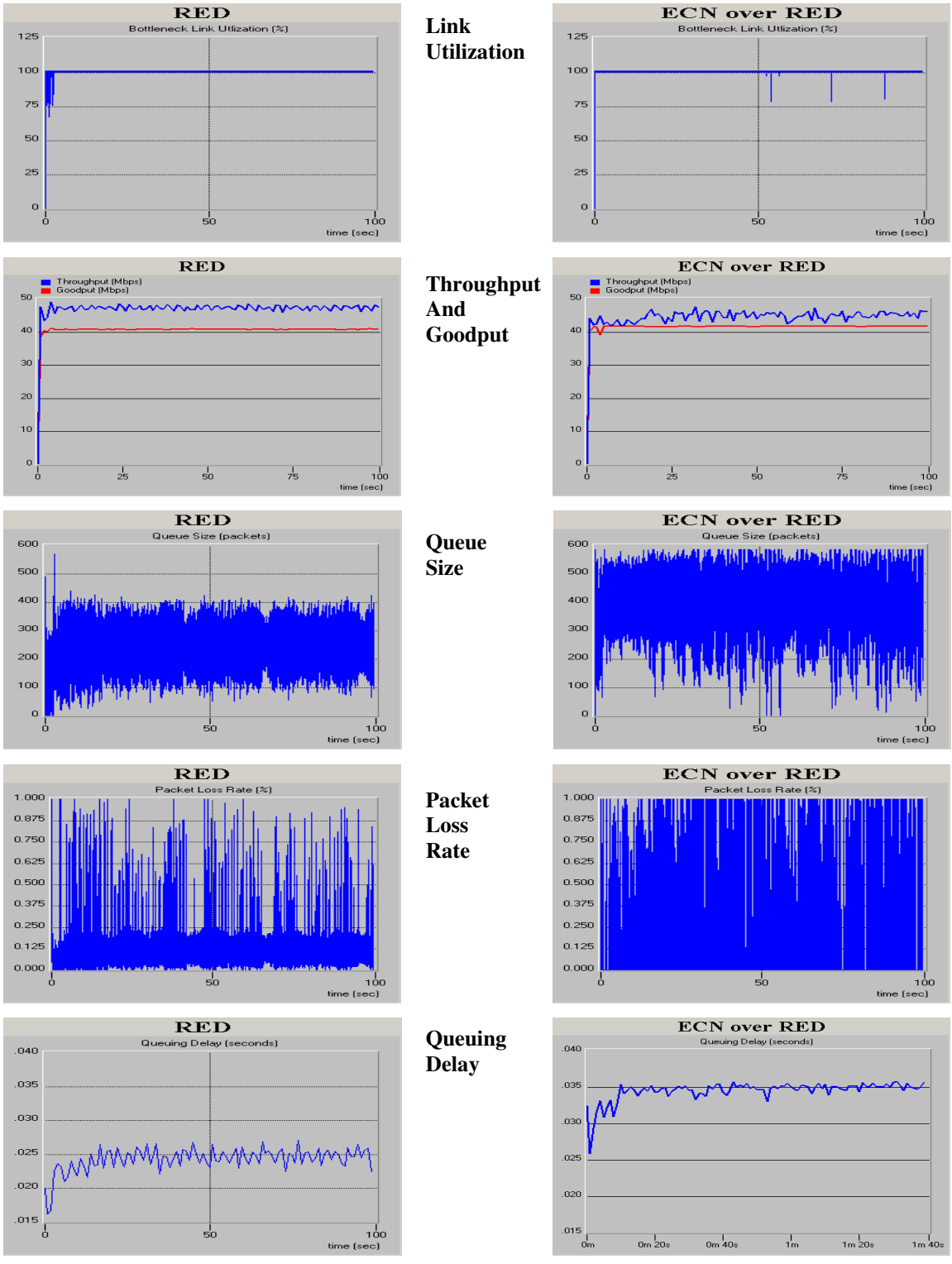
Packet loss rate for both RED and "ECN over RED" protocol is higher for 500 sources scenario compared to the 100 sources scenario. Packet drop rate is related to queue size. Packet drop rate for "ECN over RED" is higher than RED because of the higher queue size. Higher queue size and high oscillation in the queue size often results in buffer overflow that leads to higher packet loss. These graphs clearly show the effect of instability in queue size for RED and "ECN over RED" protocols.

Queuing delay for "ECN over RED" protocol is higher than the RED protocol. Also queuing delay for both RED and "ECN over RED" is higher for 500 sources scenario compared to the 100 sources scenario. This is because of the higher queue size resulting in the longer time for the packets in the router queue waiting for processing.

Figure 3-4 shows the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for RED and "ECN over RED" for 1000 sources.

Link utilization for both RED and "ECN over RED" is 100%, i.e. both the protocols are able to fully utilize the bottleneck link for 1000 sources scenario. With 1000 sources bottleneck link is better utilized (for both RED and "ECN over RED" protocols) compared to the 100 and 500 sources. This is because of the higher input date rate for the router and the bottleneck link.

"Throughput and Goodput" graphs indicate that the goodput i.e. the effective data rate for "ECN over RED" protocol is higher compared to RED protocol. These graphs also indicate that the difference between throughput and goodput, i.e. the number of retransmit packets, is quite high for RED protocol compared to the 100 sources (Figure 3-2) and 500 sources (Figure 3-3) scenarios. This is because of the higher input date rate to the router resulting in high number of packet drops to indicate congestion. Similar to 500 sources and 100 sources scenario, the difference between throughput and goodput is low for "ECN over RED" protocol for 1000 sources scenario, compared to the RED protocol. However with 1000 sources, "ECN over RED" protocol does results in small number of retransmit packets, but this number is lower than the RED protocol for same number of sources.



(a) RED **(b) ECN over RED**

Figure 3-4 RED and "ECN over RED" Performance Characteristics (1000 Sources)

Queue size for "ECN over RED" protocol is higher than the "RED" protocol. Queue size for both RED and "ECN over RED" is higher for 1000 sources scenario compared to the 100 sources and 500 sources scenario. Unstable router queue size with “RED” and “ECN over RED” protocols is quite clear from these graphs. For this scenario, the marked packets in the queue, due to congestion for "ECN over RED", result in router queue overflows and that leads to higher packet loss.

Packet loss rate for both "RED" and "ECN over RED" protocol is higher for 1000 sources scenario compared to the 100 sources and 500 sources scenarios. Packet drop rate for "ECN over RED'" is higher because of the higher queue size. Higher queue size and high oscillation in the queue size often results in buffer overflow that leads to higher packet loss.

Similar to 100 sources and 500 sources scenarios, queuing delay for "ECN over RED" protocol is higher than the RED protocol. Also queuing delay for both RED and "ECN over RED" is higher for 1000 sources scenario compared to the 100 sources and 500 sources scenarios. This is because of the higher queue size resulting in the longer time for the packets in the router queue waiting for processing.

Comparing the results of RED and “ECN over RED” for 100, 500 and 1000 sources show that:

- The bottleneck link reaches its full capacity with the increase in number of sources for RED protocol. However, the link is better utilized for “ECN over RED” protocol, all three simulation scenarios.
- The number of retransmit packets (difference between throughput and goodput) increases with the increase in the number of sources for both RED and “ECN over RED”. However, the rate of increase in retransmit packets is much lower for “ECN over RED”.
- Queue size for both RED and “ECN over RED” depends upon the number of sources i.e. active connections. As the queue size for “ECN over RED” is higher than RED, it often leads to buffer overflow especially with higher number of sources.
- Packet drop rate with RED protocol increases with the increase in the number of sources. With “ECN over RED”, the drop rate is lower than RED for small number

of sources, i.e. 100 sources scenario. However, the packet drop rate goes up significantly with the increase in the number of sources i.e. network traffic. This is because of the higher queue size with “ECN over RED’ protocol.

- Queuing delay for both RED and “ECN over RED” increases with the increase in the number of sources.

“ECN over RED” protocol directly affects the basic RED protocol by increasing the average queue size resulting in higher queuing delay. Higher queue size with ECN can push the operating queue size beyond the buffer size resulting in instability in performance and sustained steady-state errors. This behavior is clear in the simulation results of 1000 sources in Figure 3-4. For this scenario, the marked packets in the queue, due to congestion for "ECN over RED", result in router queue overflows and that leads to higher packet loss. Increasing the buffer size to work around instability and queue oscillations in the system is not the viable option since this can lead to unacceptable and unnecessary higher queuing delay. On the other hand, as the number of connections becomes small, the impact of congestion notifications increases that can result in underutilization as many sources back-off their sending rates in response to congestion notifications. With most of the static parameters (min_{th}, max_{th}) of RED and "ECN over RED" protocols directly relating to the queue size, the deviations (or errors) and oscillations in the queue size from operation points may lead to unstable system (queue overflows, link underutilization, strong synchronization among TCP connections). So there is a clear need to tune the parameters depending upon the topology and traffic loads for “ECN over RED” protocol. Various load adaptive mechanisms proposed for RED can also be applied to “ECN over RED” to enhance the effectiveness by dynamically changing the parameters as the number of connections change.

The relative advantage of “ECN over RED” decreases with the increase in number of sources i.e. network traffic. For the 1000 sources simulation scenario, Figure 3-4, the difference between goodput and throughput (number of retransmit packets) is higher than the lower traffic scenarios (Figure 3-2, Figure 3-3). However, it is still better than RED. Higher queue size with “ECN over RED”, for high network traffic load, also results in higher packet loss and queuing delay.

3.4.2 MECN over RED

We have also run the simulation and collected performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for “MECN over RED” protocol. Recall that “MECN over RED” (Section 3.1.2) differs from “ECN over RED” in that “MECN over RED” marks packets when the average quite size exceeds max_{th} and drops packets only when the router queue overflows.

Performance results for “MECN over RED” are similar to “ECN over RED” and it does not provide any significant improvement compared to “ECN over RED”. Refer to Appendix B for the performance graphs and detailed analysis.

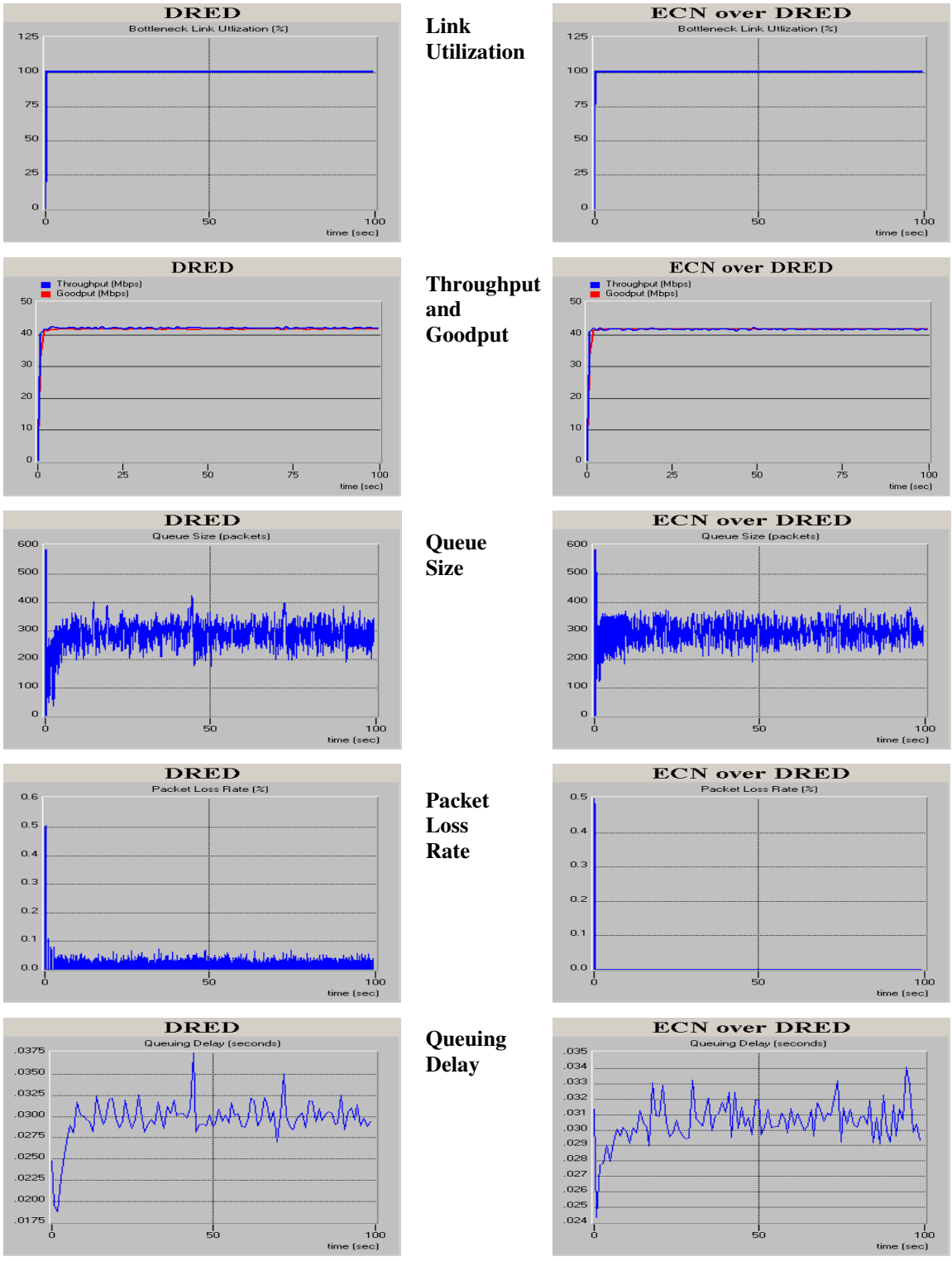
3.4.3 ECN over DRED

Figure 3-5 shows the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for DRED and “ECN over DRED” algorithms for 100 sources scenario. Note that the sources are modeled as greedy FTP connections; that is, they always have data to send as long as their congestion window permits.

The “Link Utilization” graphs in Figure 3-5 indicate the bottleneck link utilization in our network model for DRED and "ECN over DRED" protocols. Bottleneck link should be optimally utilized for better network performance. Both DRED and “ECN over DRED” result in 100 % utilization of this link

The "Throughput and Goodput" graphs in Figure 3-5 show throughput and goodput performance characteristics for DRED and "ECN over DRED" protocols for 100 TCP sources scenario. Refer to Section 2.3 for description of these performance characteristics. These Graphs indicate that throughput and goodput for "ECN over DRED" is same as DRED for this scenario. Note that the difference between throughput and goodput represents the number of retransmit packets. For the 100 sources scenario, total number of retransmit packets is quite low for both DRED and “ECN over DRED”.

The "Queue Size" graphs in Figure 3-5 indicate the router queue size. Queue size for DRED and “ECN over DRED” is same for 100 sources scenario. Note that the queue size is quite stable around the target level (293 packets) for both the protocols.



(a) DRED **(b) ECN over DRED**

Figure 3-5 DRED and "ECN over DRED" Performance Characteristics (100 Sources)

The "Packet Drop Rate" graphs in Figure 3-5 show the number of packets dropped by the router due to congestion (for "DRED") and queue overflow (for both DRED and "ECN over DRED"). These graphs indicate that the packet drop rate is higher for DRED protocol compared to "ECN over DRED". As the "Queue Size" graphs indicate that there is no router queue overflow for 100 sources scenario, the Packet Drop Rate for "ECN over DRED" protocol is 0. Packets are dropped by DRED protocol for congestion control (to indicate congestion).

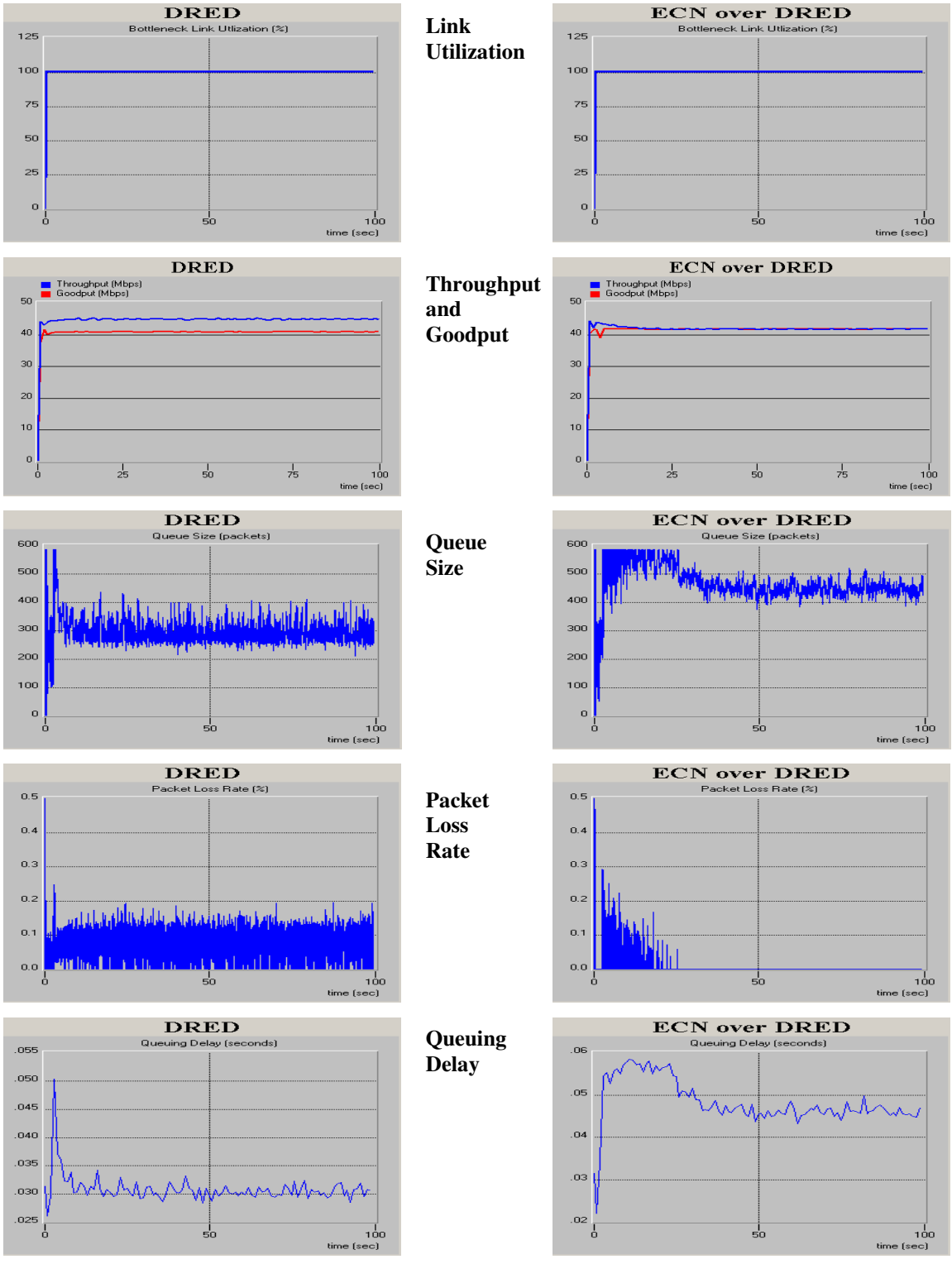
The "Queuing Delay" graphs in Figure 3-5 indicate the router queuing delay for 100 sources scenario. There is no significant different in the queuing delay for both DRED and “ECN over DRED” protocols.

Figure 3-6 shows the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for DRED and “ECN over DRED” for 500 sources.

Similar to 100 sources scenario, bottleneck link is optimally utilized (100%) for both DRED and “ECN over DRED” algorithms.

The "Throughput and Goodput" graphs indicate that the goodput i.e. the effective data rate for "ECN over DRED" protocol is slightly higher compared to DRED protocol. These graphs also indicate that the difference between throughput and goodput, i.e. the number of retransmit packets, is quite high for DRED protocol for 500 sources scenario compared to the 100 sources scenario (Figure 3-5). This is because of the higher input date rate to the router resulting in high number of packet drops to indicate congestion. The difference between throughput and goodput is quite low for "ECN over DRED" protocol. This is a major advantage of "ECN over DRED" protocol as it results in lower number of retransmissions and higher data rate.

Queue size for "ECN over DRED" protocol is higher than the DRED protocol. Queue size for “DRED” protocol is same for both 100 and 500 sources scenario, which is close to the target queue size (refer Section 2.2.2). However, queue size for “ECN over DRED” is higher than the target level because of the extra packets, which are marked and queued instead of dropped to indicate congestion.



(a) DRED **(b) ECN over DRED**

Figure 3-6 DRED and "ECN over DRED" Performance Characteristics (500 Sources)

Packet loss rate for both DRED and "ECN over DRED" protocol is higher for 500 sources scenario compared to the 100 sources scenario. Packet drop rate for "ECN over DRED'" is higher because of the higher queue size. Higher queue size results in buffer overflow, especially at the start of the simulation, which leads to packet loss.

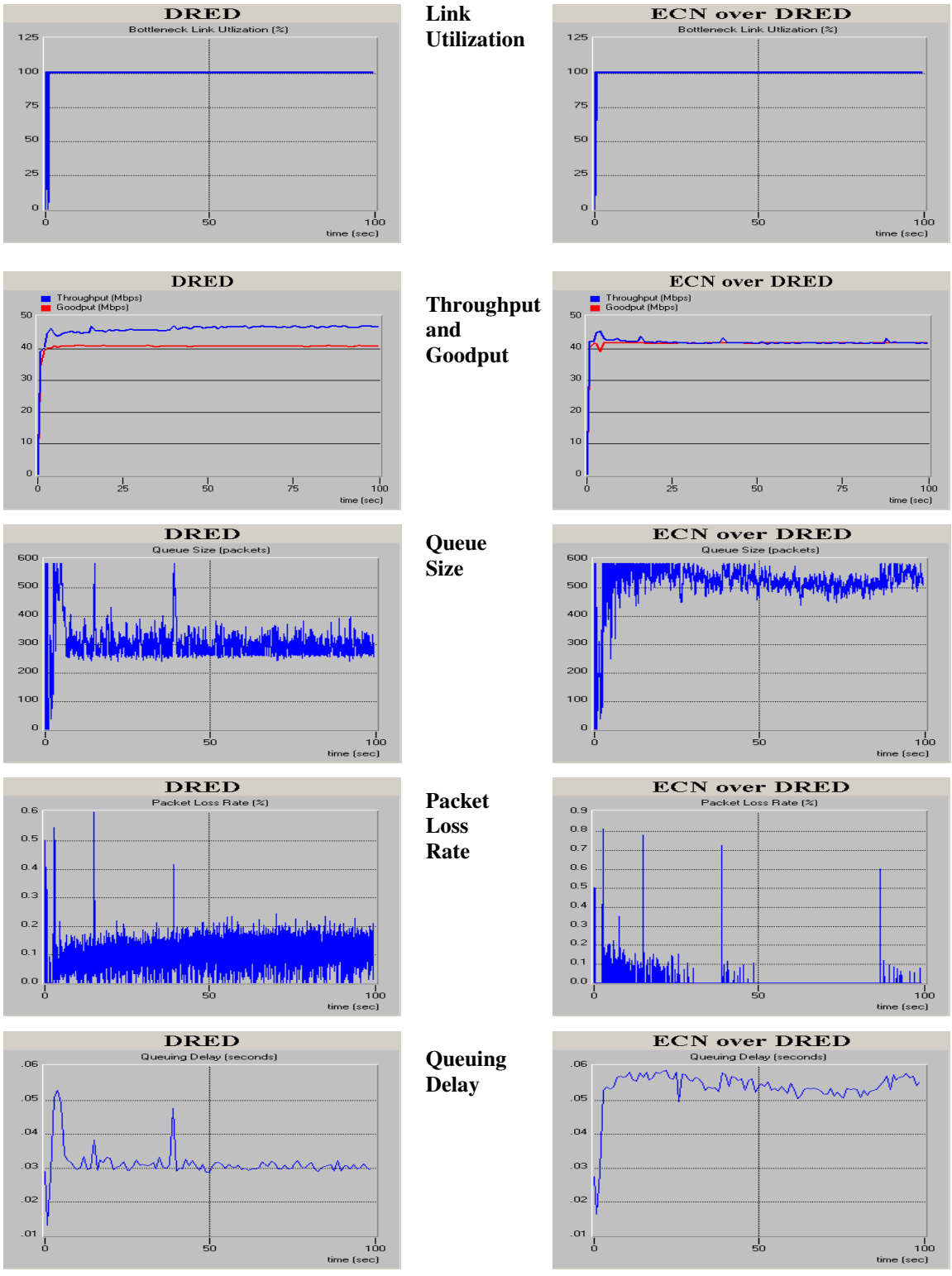
Queuing delay for "ECN over DRED" protocol is higher than the DRED protocol. This is because of the higher queue size resulting in longer time for the packets in the router queue waiting for processing. Queuing delay for DRED is same for both 100 sources (Figure 3-5) and 500 sources (Figure 3-6) scenarios. This is because of the stable queue size with DRED for different network traffic loads.

Figure 3-7 shows the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for DRED and “ECN over DRED” for 1000 sources.

Similar to 100 sources and 500 sources scenario, bottleneck link is optimally utilized (100%) for both DRED and “ECN over DRED” algorithms.

The "Throughput and Goodput" graphs indicate that the goodput i.e. the effective data rate for "ECN over DRED" protocol is slightly higher compared to DRED protocol. These graphs also indicate that the difference between throughput and goodput, i.e. the number of retransmit packets, is quite high for DRED protocol for 1000 sources scenario compared to the 100 sources (Figure 3-5) and 500 sources (Figure 3-6) scenarios. The difference between throughput and goodput is quite low for "ECN over DRED" protocol. This is a major advantage of "ECN over DRED" protocol as it results in lower number of retransmissions and higher data rate.

Queue size for "ECN over DRED" protocol is higher than the DRED protocol. Queue size for DRED protocol is same for all three sources scenarios (100, 500 and 1000 sources), and is also close to the target queue size (refer Section 2.2.2). However, queue size for “ECN over DRED” is higher than the target level because of the extra packets, which are marked and queued instead of dropped to indicate congestion. Queue size for both DRED and “ECN over DRED” protocols is stable and does not oscillate to the level we have seen with the RED and “ECN over RED” protocols.



(a) DRED **(b) ECN over DRED**

Figure 3-7 DRED and "ECN over DRED" Performance Characteristics (1000 Sources)

Graphs show that the packet loss rate for “ECN over DRED” is lower than the DRED protocol because the packets are marked and queued with “ECN over DRED” instead of dropped as in DRED protocol. Packet loss rate for both DRED and "ECN over DRED" protocol is higher for 1000 sources scenario compared to the 100 sources and 500 sources scenario. Packet drop rate for "ECN over DRED'" is higher because of the higher queue size. Higher queue size often results in buffer overflow, which leads to packet loss.

Queuing delay for "ECN over DRED" protocol is higher than the DRED protocol. Note that the queuing delay with DRED does not change with the change in traffic load. However, this is not true for “ECN over DRED”. These graphs show that the queuing delay for 1000 sources scenario is higher than 500 sources and 100 sources scenario. This is because of the higher queue size resulting in longer time for the packets in the router queue waiting for processing.

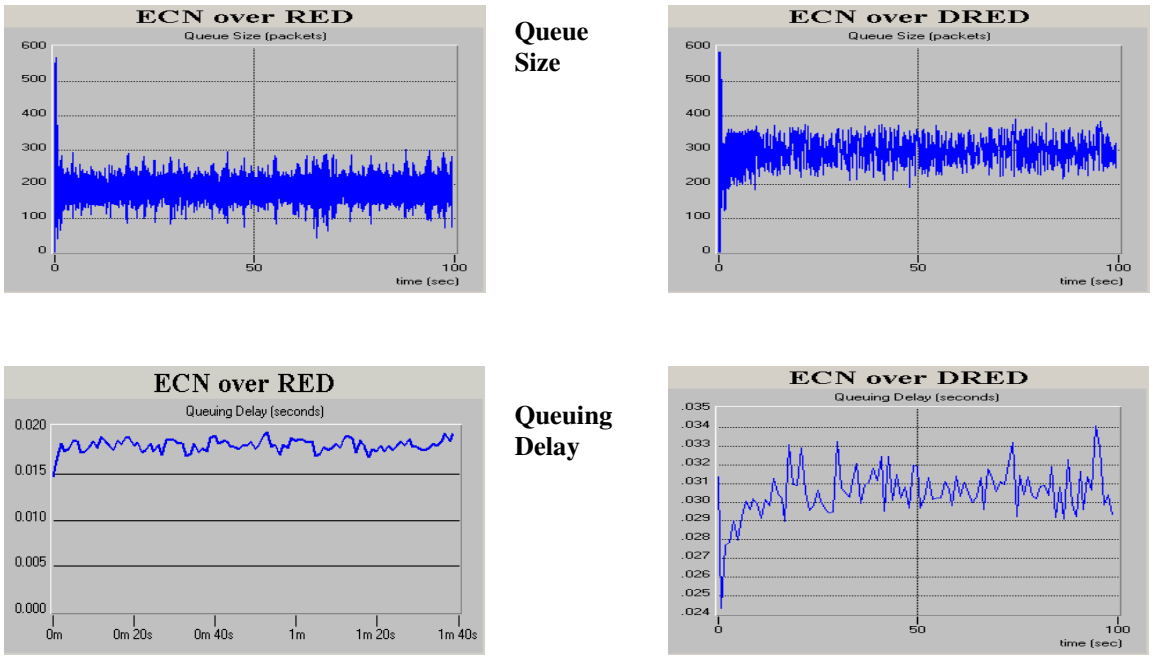
Comparing the results of DRED and “ECN over DRED” for 100, 500 and 1000 sources show that:

- Both DRED and “ECN over DRED” result in 100 % utilization of this link for all three simulation scenarios i.e. 100, 500 and 1000 sources.
- For both DRED and “ECN over DRED”, the difference between throughput and goodput, i.e. the number of retransmits packets, is highest for the 1000 sources scenario, followed by the 500 sources scenario and lowest for the 100 sources scenario. However this rate of increase is much lower for “ECN over DRED” compared to the DRED protocol.
- Goodput, i.e. the effective date rate, is higher for “ECN over DRED” protocol compared to the DRED protocol for all three simulation scenarios (100, 500 and 1000 sources).
- DRED is effective at stabilizing and keeping the queue size around the control target, as the queue size is same for all three simulation scenarios i.e. 100, 500 and 1000 sources. Queue Size with “ECN over DRED” increases with the increase in number of sources. This is because of the higher number of packets that are marked for congestion and queued with “ECN over DRED”. These kinds of packets increase

with the increase in the number of sources i.e. traffic load. The stable and predictable queue size is a major advantage of DRED over RED. This advantage is clearly lost with “ECN over DRED” because of the increase in queue size with the increase in network traffic. The results show that although the queue size is stable (non-oscillating) for “ECN over DRED”, it does increase with the increase in network traffic (number of sources).

- Packet drop rate with DRED protocol increases with the increase in the number of sources. This is because DRED protocol adaptively computes drop probabilities as the network traffic load increases or decreases to maintain the queue size. Packet drop rate also increases for “ECN over DRED” with the increase in the number of sources. This is because of the higher queue size (for high traffic load) that often results in queue overflow leading to packet loss.
- Queuing delay for DRED is same for all the three simulation scenarios (100, 500 and 1000 connections). This is because of the stable queue size for different network traffic loads. The stable and predictable queuing delay is a major advantage of DRED over RED. This advantage is somewhat lost with “ECN over DRED”, as its queuing delay increases with the increase in traffic load i.e. number of sources.

“ECN over DRED” protocol directly affects the basic DRED protocol by increasing the average queue size resulting in buffer overflows, which leads to higher packet loss rate and higher queuing delay. Predictable queue size around the control target is a major advantage is DRED over RED. This advantage is clearly lost with “ECN over DRED” because of the increase in queue size with the increase in network traffic. Higher queue size with ECN can push the operating queue size beyond the buffer size resulting in instability in performance and sustained steady-state errors. This behavior is clear in the simulation results of 1000 sources scenario in Figure 3-7. For this scenario, the marked packets in the queue, due to congestion for "ECN over DRED" protocol, result in frequent router queue overflows resulting in higher packet loss. Increasing the buffer size to work around instability in the system is not the viable option since this can lead to unacceptable and unnecessary higher queuing delay.



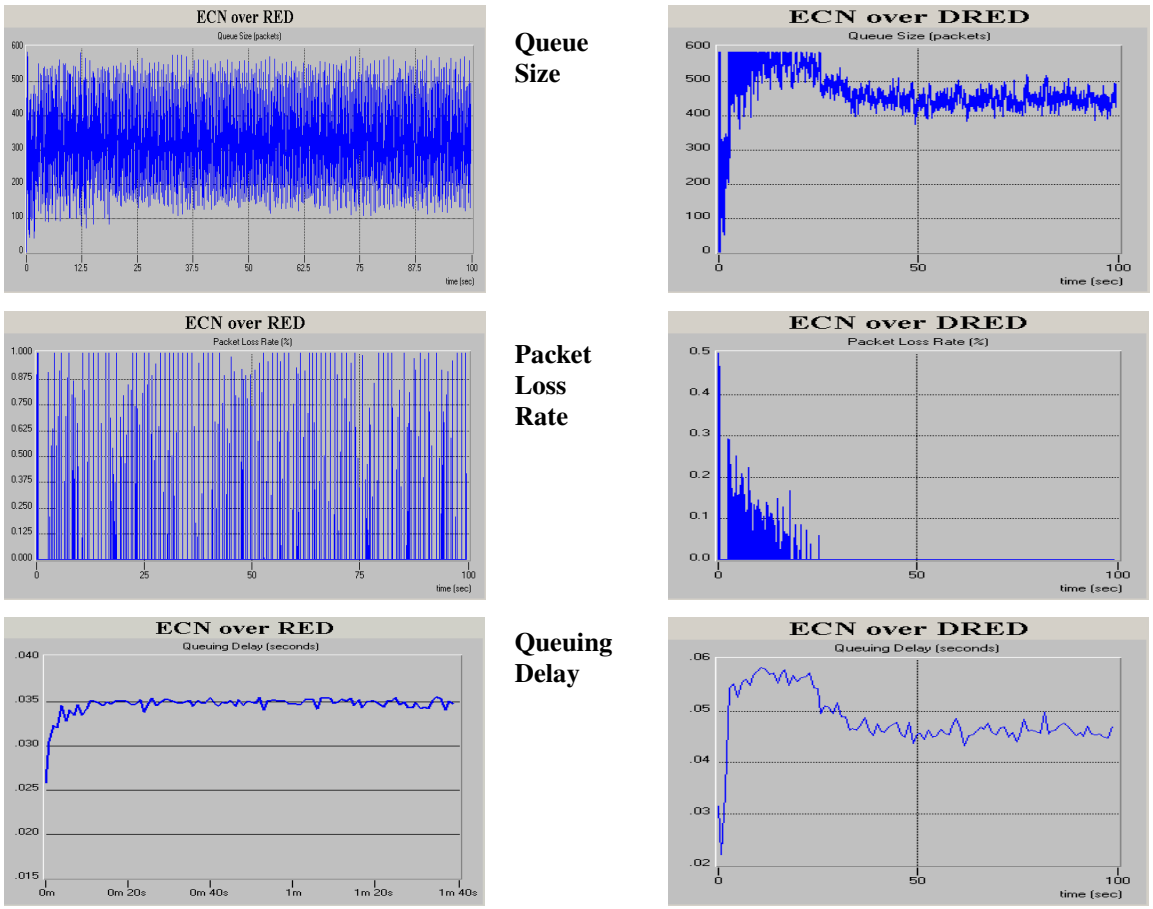
(a) ECN over RED **(b) ECN over DRED**

Figure 3-8 "ECN over RED" and "ECN over DRED" Performance Characteristics (100 Sources)

3.4.4 "ECN over RED" vs. "ECN over DRED"

In this section, we compare the results of "ECN over RED' and "ECN over DRED" protocols. Simulation results presented in Section 3.4.1 and Section 3.4.3 are used for performance analysis in this section.

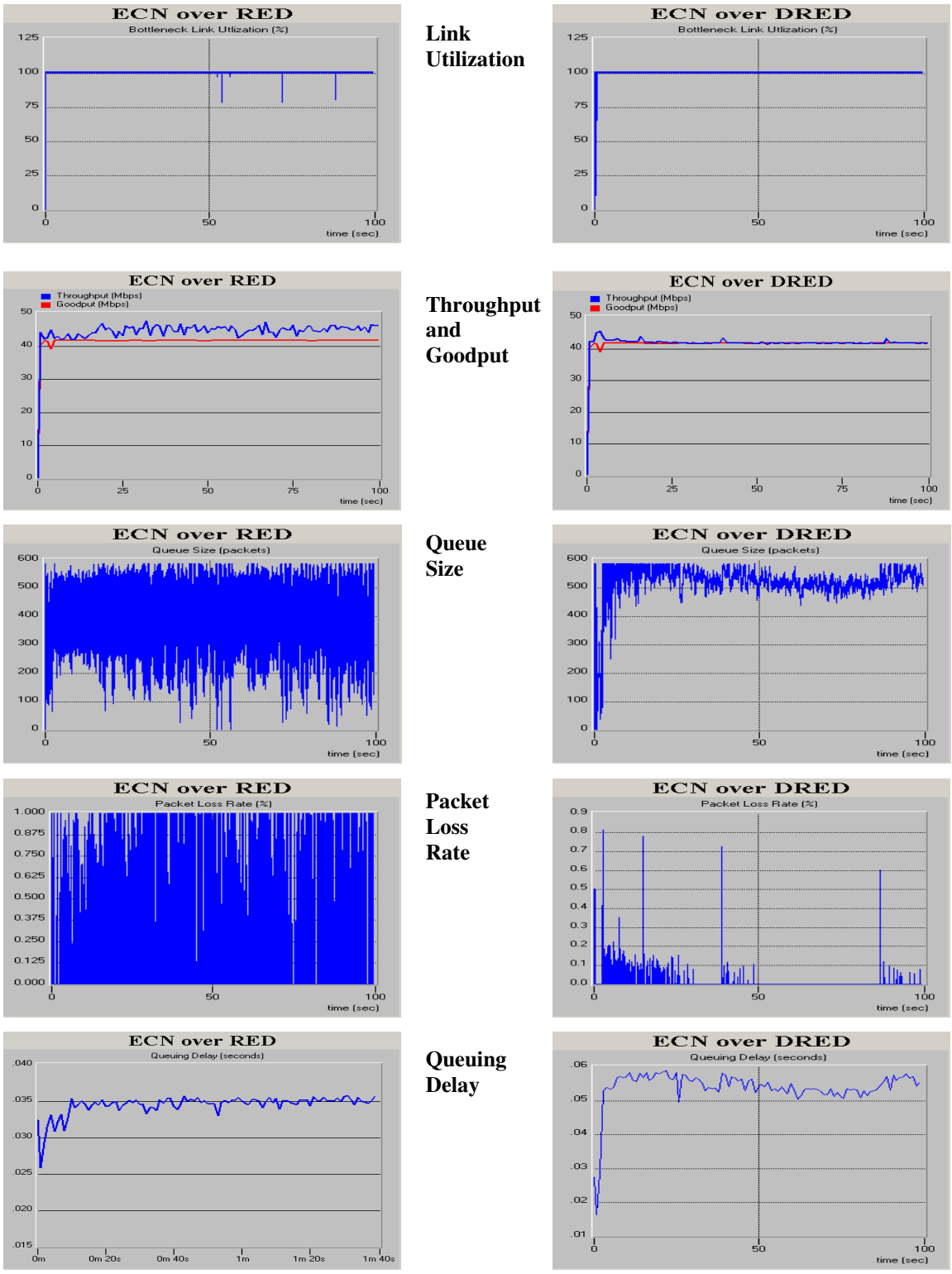
Figure 3-8 shows the performance metrics (queue size and queuing delay) for "ECN over RED" and "ECN over DRED" for 100 sources. Bottleneck link utilization, throughput, goodput (i.e. effective data rate), packet loss rate is similar for both "ECN over RED" and "ECN over DRED" for 100 sources scenario (refer to Figure 3-2 and Figure 3-5 for graphs). "ECN over DRED" is effective at stabilizing the queue size around the control target. Queue size with "ECN over RED" is also quite stable for 100 sources scenario. There is no packet loss for these two protocols with 100 sources. As the queue size for "ECN over DRED' is higher than "ECN over RED", the queuing delay for "ECN over DRED" is higher than "ECN over RED".



(a) ECN over RED **(b) ECN over DRED**

Figure 3-9 "ECN over RED" and "ECN over DRED" Performance Characteristics (500 Sources)

Figure 3-9 shows the performance metrics (queue size, packet loss rate, queuing delay) for "ECN over RED" and "ECN over DRED" for 500 sources scenario. Queue size for "ECN over RED" oscillates violently around the thresholds and is quite unstable. Queue size for "ECN over DRED" is higher than the control target because of the packets that are marked and queued to indicate congestion. However, queue size with "ECN over DRED" is stable (non-oscillating). Packet loss rate is higher for "ECN over RED" compared to "ECN over DRED", this is because of the unstable queue size with "ECN over RED" resulting in frequent buffer overflows. Queuing delay for "ECN over DRED" is higher than "ECN over RED"; this is because of the higher queue size with "ECN over DRED".



(a) ECN over RED **(b) ECN over DRED**

Figure 3-10 "ECN over RED" and "ECN over DRED" Performance Characteristics (1000 Sources)

Figure 3-10 shows the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for “ECN over RED” and “ECN over DRED” for 1000 sources.

Similar to 100 sources and 500 sources scenario, bottleneck link is optimally utilized (100%) for both “ECN over RED” and “ECN over DRED” algorithms.

The difference between goodput and throughput, i.e. the number of retransmit packets is low for “ECN over DRED” when compared to “ECN over RED”. However, the effective data rate is same for both these protocols.

Queue size for “ECN over RED” oscillates violently and often crosses the queue buffer size for 1000 sources scenario. “ECN over DRED” protocol results in higher than target queue size because of the congestion marked packets in the queue. However, queue size if quite stable (non-oscillating) compared to “ECN over RED”.

Packet loss rate is quite high for “ECN over RED” compared to the “ECN over DRED”. This is again because of the unstable queue size with “ECN over RED” which results in buffer overflow leading to packet loss. Packet loss rate for 1000 sources scenario is higher than the 100 and 500 sources scenario for both “ECN over RED” and “ECN over DRED”. However, the rate of increase with the increase in number of sources is much lower for “ECN over DRED” compared to “ECN over RED”.

Queuing delay for “ECN over DRED” is higher than “ECN over RED”; this is because of the higher queue size with “ECN over DRED”.

These results clearly indicate that queue size for “ECN over RED” depends upon the number of sources i.e. active connections. DRED is effective at stabilizing and keeping the queue size around the control target (refer Section 3.1.3). This is the major advantage of DRED over RED. However, the queue size with “ECN over DRED” is higher than the control target. This is because of the packets are marked and queued instead of dropped as in DRED protocol. So the queue size with “ECN over DRED” depends upon the traffic load, and the DRED advantage of predictable queue size around the target level is lost. Increasing the buffer size to work around this problem is not viable as this can lead to unacceptable and unnecessary higher queuing delay. However, queue size with “ECN over DRED” is quite stable and does not oscillate as with “ECN over RED”.

3.4.5 Fairness

This section describes the performance of different protocols (RED, "ECN over RED", DRED and "ECN over DRED") for heterogeneous networks i.e. when TCP connections have different RTTs. The basic behavior of TCP has bias towards the shorter connections (connections with lower RTTs) when the connections have different RTTs, resulting in higher goodput for shorter connections [Floy91a]. The objective of this section is to study the effect of RTT for RED, "ECN over RED", DRED, and "ECN over DRED" protocols using Jain's Fairness Index and Visual Max-Min Fairness [Section 2.3.7].

The study uses the bottleneck network configuration shown in Figure 2-6. The network has a configurable number of subnets with 100 sources in each subnet. For fairness evaluation, the RTT is set differently for connections from different subnets. However, the RTT is equal for all the connections from a particular subnet. Both the techniques used in this study for fairness evaluation (Jain's Fairness Index and Visual Max-Min Fairness) require goodput. So Average Goodput of each subnet is calculated during the simulation.

3.4.5.1 RED and "ECN over RED"

Jain's Fairness Index

Jain's fairness index postulates that the network is a multi-user system, and drives the metrics to see how fairly each user is treated. For this study each subnet is equivalent to a user. Jain's Fairness Index requires the total goodput for each subnet, and is calculated as for a set of subnet (n) goodput ($x_1, x_2, ..., x_n$):

$$f(x_1, x_2, ..., x_n) = (\sum_{i=1}^{n} x_i)^2 / (n * \sum_{i=1}^{n} x_i^2)$$

Simulation is run for 100 seconds to calculate the goodput for each source TCP. Note that the data collected during the first 20 seconds of simulation time is discarded to reduce the startup and stabilization effect. Goodput for all the sources in a particular subnet are added to calculate the total goodput for the subnet.

Table 3-3 shows the Jain's Fairness Index for RED and "ECN over RED" for 3 subnets (300 sources) and 5 subnet (500 sources) scenarios:

Table 3-3 Jain's Fairness Index: RED and "ECN over RED"

	RED	"ECN over RED"
3 Subnets (300 Sources) Subnet 0 RTT: 60 msec. Subnet 1 RTT: 100 msec Subnet 2 RTT: 140 msec	0.9364	0.8688
5 Subnets (500 Source) Subnet 0 RTT: 60 msec Subnet 1 RTT: 80 msec Subnet 2 RTT: 100 msec Subnet 3 RTT: 120 msec Subnet 4 RTT: 140 msec	0.9124	0.8838

For the 3-subnets scenario, Jain's Fairness Index for RED is 0.9364 and for "ECN over RED" is 0.8688. For the 5-subnets scenario, Jain's Fairness Index for RED is 0.9124 and for "ECN over RED" is 0.8838. Note that the Jain's Fairness Index value of 1 indicates that all the flows get exactly the same goodput i.e. the user data rate.

Since the perfect fairness has a Jain's fairness index of 1, it is clear that RED is fairer than "ECN over RED" for both 3 subnets and 5 subnets scenarios. The 3-subnets scenario indicates the effect of large changes in RTT (40 msec) and the 5-subnets scenario indicates the effect of small changes in RTT (20 msec). Jain's Fairness index number from Table 3-3 shows that RED algorithm is fairer for 3-subnets scenario compared to the 5-subnets scenario. However, "ECN over RED" is fairer for 5-subnets scenario compared to the 3-subnets scenario. With ECN fairness is hit quite severely for the 3-subnets scenario, compared to the 5-subnets scenario, which shows the impact of large different in RTT between competing connections. These numbers also indicate that the impact on fairness for the RED and "ECN over RED" protocols depends upon the various factors (traffic load, network configuration) and is not predictable.

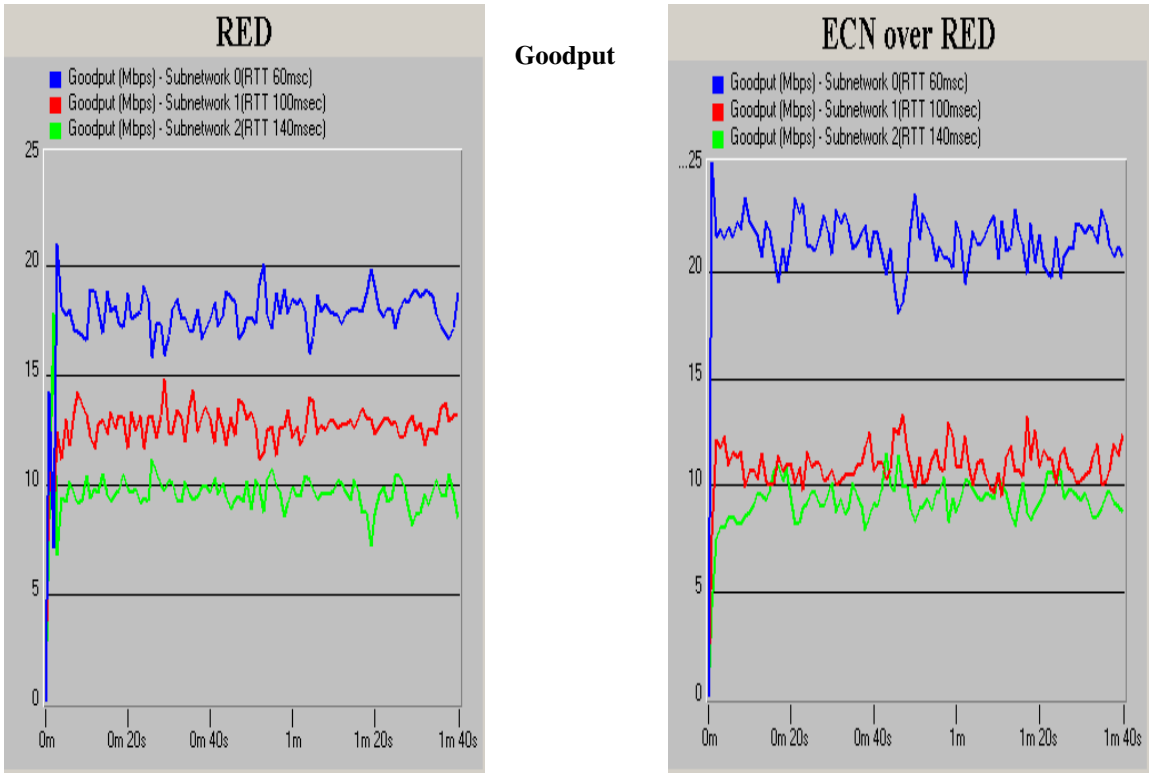


Figure 3-11 RED and "ECN over RED" Goodput (300 Sources)

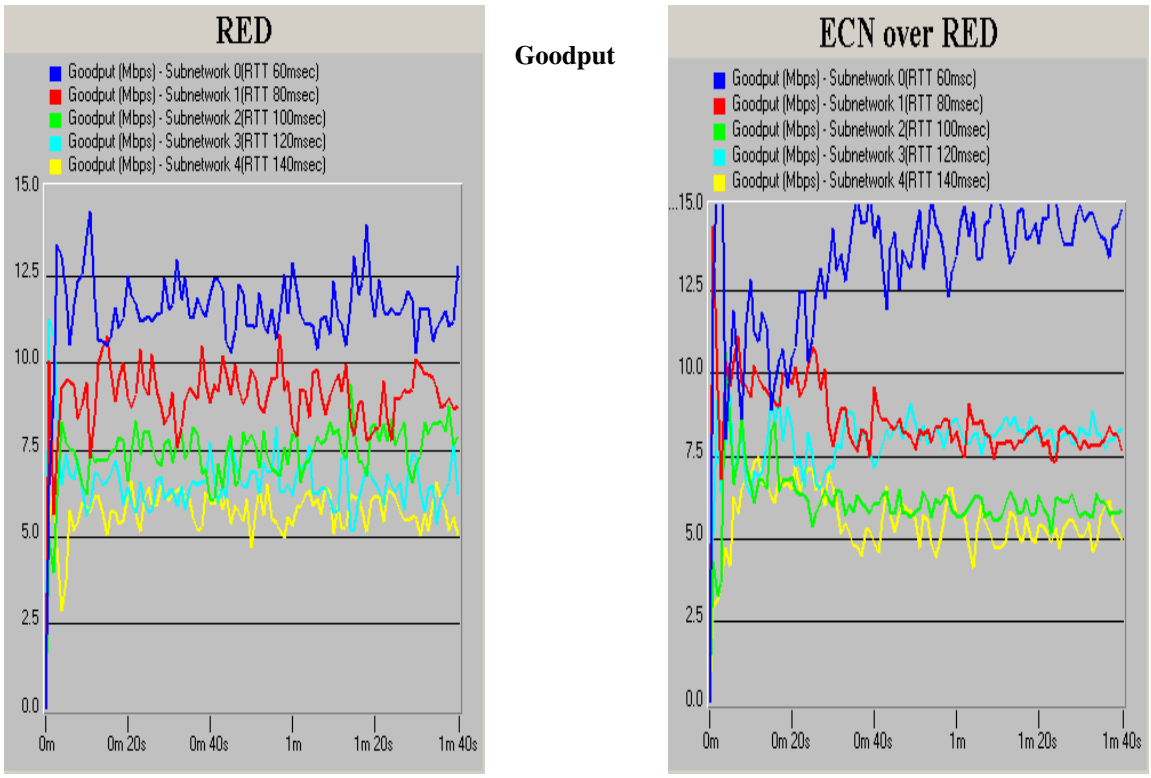


Figure 3-12 RED and "ECN over RED" Goodput (500 Sources)

Visual Max-Min Fairness

Figure 3-11 and Figure 3-12 show the graphs to visually analyze the max-min fairness with respect to goodput (i.e. effective data rate) for different subnets with different RTTs. Note that the goodput of a subnet represents the total goodput from all the sources in that particular subnet.

Figure 3-11 provides a visual sense of max-min fairness via the difference among the goodput of three subnets with different RTTs (60msec, 100msec and 140msec), for RED and “ECN over RED”. The graphs clearly show that the goodput of an individual connection depends largely on its round-trip time, and there is a bias towards shorter connections (Connections with lower RTTs) when the connections have different RTTs. In these graphs, the average goodput of the sources in Subnet-0 (with lowest RTT - 60 msec) is highest, followed by that of Subnet-1 (RTT – 100msec) and is lowest for Subnet-2 (RTT - 140 msec). Therefore there is a bias towards the shorter connections for both "RED" and "ECN over RED". From the difference between goodput of subnets with different RTTs, these graphs show that RED is fairer than “ECN over RED”. This observation confirms the Jain’s Fairness Index calculations in the previous section. The goodput for the connections with lower RTT increases and the goodput for the connections with higher RTT decrease resulting in unfairness.

Figure 3-12 provides a visual sense of max-min fairness via the difference between goodput for five subnets with different RTTs (60msec, 80msec, 100msec, 120msec, 140msec) for RED and “ECN over RED”. Note that the incremental increase in RTT from the lowest RTT connections is 20 msec in this scenario compared to 40 msec in the 3 subnet scenario discussed above. This scenario shows the impact on goodput with small changes in RTT values (20 msec) of completing connections, compared to the large changes (40 msec) discussed above (Figure 3-11). These graphs clearly show the significant impact on Goodput for the small changes in RTT. The Goodput of an individual connection depends largely on its round-trip time and there is a bias towards shorter connections (lower RTT), for both "RED" and "ECN over RED", similar to the results of 3 subnet scenarios (Figure 3-11). By comparing the graphs from Figure 3-11 and Figure 3-12, we observe that the impact on goodput is proportional to the difference in RTTs of competing connections.

3.4.5.2 DRED and “ECN over DRED”

Jain's Fairness Index

Jain’s fairness index postulates that the network is a multi-user system, and drives the metrics to see how fairly each user is treated. Refer to Section 2.3.7.1 for detailed description of this algorithm. The following table shows the Jain’s Fairness Index for DRED and “ECN over DRED”:

Table 3-4 Jain's Fairness Index: DRED and "ECN over DRED"

	DRED	“ECN over DRED”
3 Subnets (300 Sources) Subnet 0 RTT: 60 msec. Subnet 1 RTT: 100 msec Subnet 2 RTT: 140 msec	0.9558	0.9458
5 Subnets (500 Source) Subnet 0 RTT: 60 msec Subnet 1 RTT: 80 msec Subnet 2 RTT: 100 msec Subnet 3 RTT: 120 msec Subnet 4 RTT: 140 msec	0.9602	0.9488

For 3-subnets, Jain’s Fairness Index for DRED is 0.9558 and for “ECN over DRED” is 0.9458. For 5-subnets, Jain’s Fairness Index for DRED is 0.9602 and for “ECN over DRED” is 0.9488. Note that the Jain’s Fairness Index value of 1 indicates that all the flows get exactly the same goodput i.e. the user data rate.

Since the perfect fairness has a Jain’s fairness index of 1, it is clear that DRED is fairer than “ECN over DRED” for both 3-subnets and 5-subnets scenario. The 3-subnets scenario indicates the effect of large changes in RTT (40 msec) and the 5-subnets scenario indicates the effect of small changes in RTT (20 msec). However the decrease in fairness in “ECN over DRED” is not as severe as “ECN over RED” (Table 3-3). It is clear from these numbers that DRED and “ECN over DRED” is fairer than RED and “ECN over RED”. These numbers also indicate that the fairness for the DRED and "ECN over DRED" is stable and does not depend upon the network traffic as for RED and “ECN over RED”.

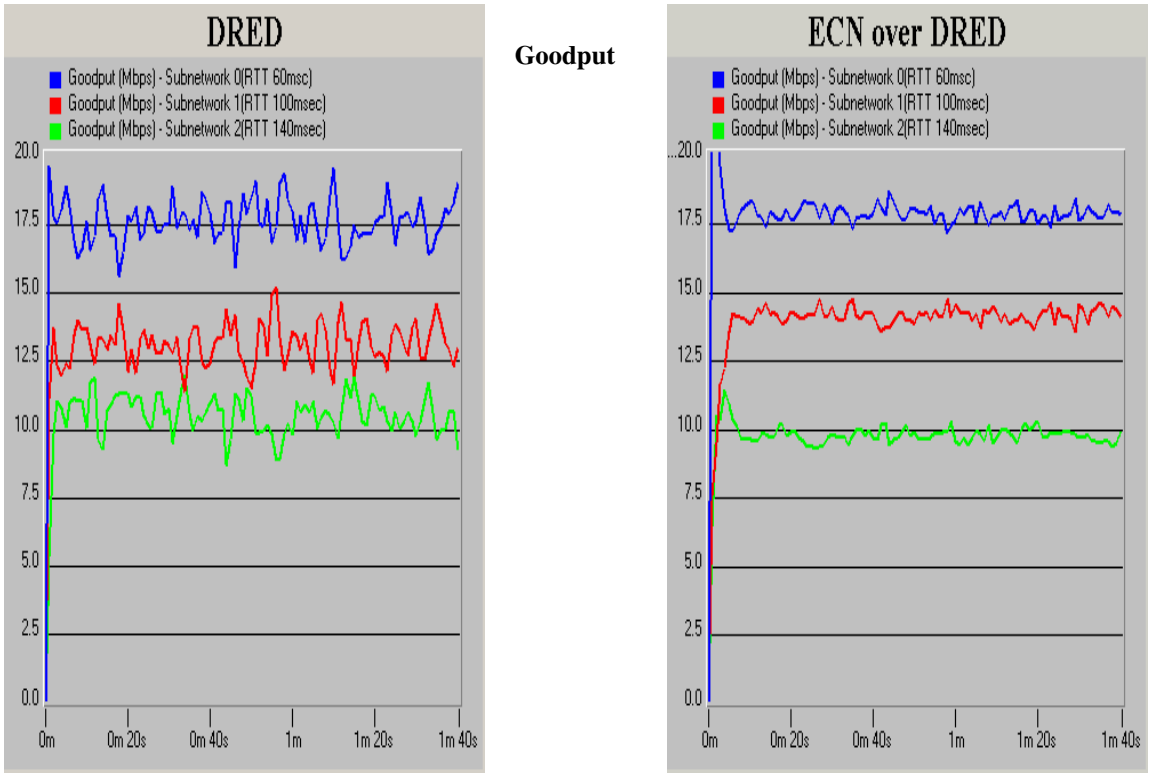


Figure 3-13 DRED and "ECN over DRED" Goodput (300 Sources)

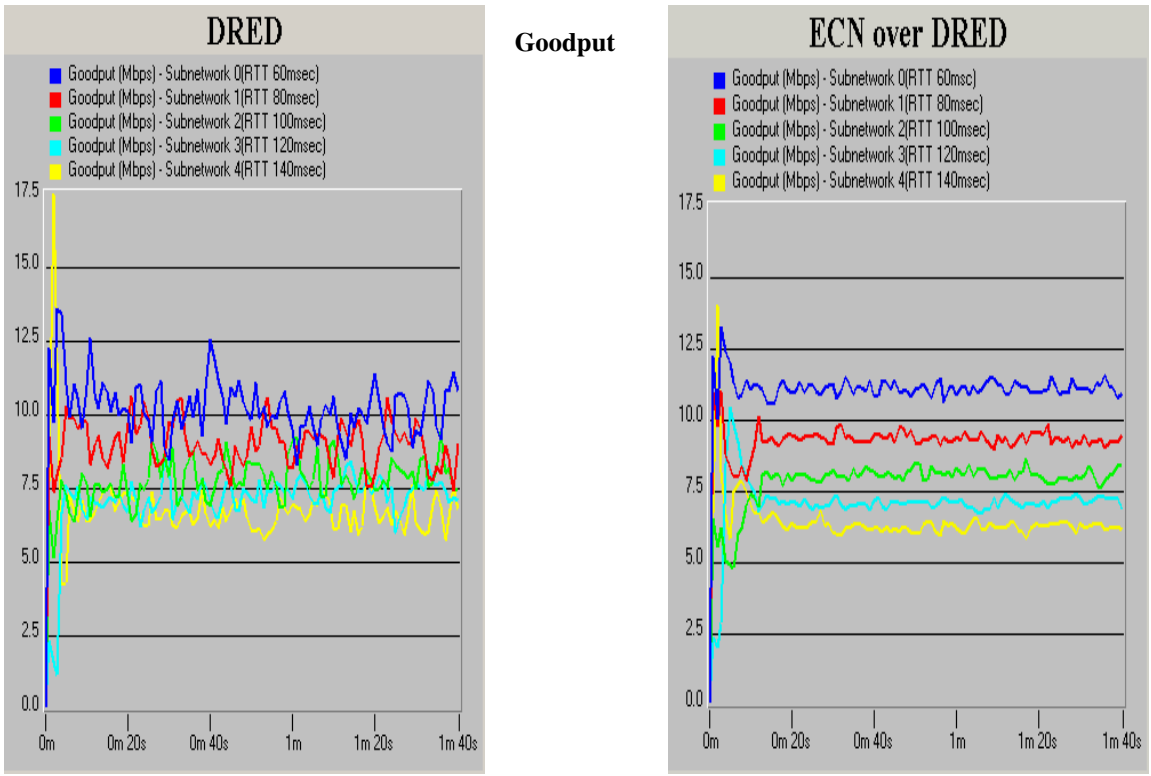


Figure 3-14 DRED and "ECN over DRED" Goodput (500 Sources)

Visual Max-Min Fairness

Figure 3-13 and Figure 3-14 show the graphs to visually analyze the max-min fairness with respect to goodput (i.e. effective data rate) for different subnets with different RTTs. Note that the goodput of a subnet represents the total goodput from all the sources in that particular subnet.

Figure 3-13 provides a visual sense of max-min fairness via the difference among the goodput of three subnets with different RTTs (60msec, 100msec and 140msec), for DRED and "ECN over DRED". The graphs clearly show that the goodput of an individual connection depends largely on its round-trip time, and there is a bias towards shorter connections (connections with lower RTTs), when the connections have different RTTs. In these graphs the average goodput of the sources in Subnet-0 (with lowest RTT - 60 msec) is highest, followed by that of Subnet-1 (RTT – 100msec) and is lowest for Subnet-2 (RTT - 140 msec). The graphs show the bias towards the shorter connections (lower RTT) for both "DRED" and "ECN over DRED". From the difference between goodput of subnets, the graphs show that DRED is fairer than "ECN over DRED". This observation confirms the Jain's Fairness Index calculations in the previous section. The goodput for the connections with lower RTTs increases and the goodput for the connections with higher RTT decreases resulting in unfairness.

Figure 3-14 provides a visual sense of max-min fairness via the difference between goodput for five subnets with different RTTs (60msec, 80msec, 100msec, 120msec, 140msec), for DRED and "ECN over DRED". Note that the incremental increase in RTT from the lowest RTT connections is 20 msec in this scenario compared to 40 msec in the 3 subnet scenario discussed above. This scenario shows the impact on goodput with small changes in RTT values (20 msec) of completing connections, compared to the large changes (40 msec) discussed above (Figure 3-13). These graphs clearly show the significant impact on Goodput for the small changes in RTT. The Goodput of an individual connection depends largely on its round-trip time and there is a bias towards shorter connections (lower RTT), for both "DRED" and "ECN over DRED", similar to the results of 3 subnet scenarios. By comparing the graphs from Figure 3-13 and Figure 3-14, we can conclude that the impact on goodput is proportional to the difference in RTTs of competing connections.

3.5 Concluding Remarks

This chapter presented the algorithms for “ECN over RED”, "MECN over RED" and “ECN over DRED” protocols, followed by a series of simulations using OPNET Modeler that evaluates the behavior and performance of these protocols.

The results show that the addition of ECN does provide better goodput, i.e. the effective data rate, for both RED and DRED. The difference between throughput and goodput, i.e. the number of retransmitted packets, are also lower for both "ECN over RED" and "ECN over DRED". The Queue size of RED and "ECN over RED" depends on the number of sources i.e. active queue connections. Queue size for both "ECN over RED" and “ECN over DRED” is also higher than RED and DRED respectively. Higher queue size can push the operating queue size beyond the buffer size resulting in frequent overflows that leads to instability in performance and sustained steady-state errors. DRED is effective at stabilizing the queue around the target value. This advantage is lost with “ECN over DRED” as the queue size increases with the network traffic load.

The TCP/IP network fairness results (using Jain’s Fairness Index and Visual Max-Min Fairness) show that neither of these protocols (RED, “ECN over RED”, "MECN over RED", DRED, “ECN over DRED”) is fair when TCP connections with different RTTs are competing for the resources. Based on these results, a fair version of ECN (FECN) is proposed in the next chapter.

4 ECN Enhancements

In the last chapter, simulation results of “ECN over RED” show that addition of ECN to RED does provide better goodput and lower retransmissions. However, the results also demonstrate that ECN also results in higher queue size and higher queuing delay. ECN adds to unfairness in heterogeneous networks with a strong bias towards connections with lower RTT. This chapter presents the following two enhancements to the “ECN over RED” to improve the performance:

- “Fair ECN (FECN) over RED” enhances the “ECN over RED” algorithm to provide better fairness for competing heterogeneous connections (TCP connections with different RTTs) by dynamically adjusting the control parameters based on the RTTs and traffic load.
- “Mark-First Fair ECN (MFFECN) over RED” further enhances the “FECN over RED” by marking the first unmarked packet in front of the queue, instead of the new packet in “ECN over RED” and “FECN over RED”, for congestion notifications.

These new algorithms, simulation model, simulation results and performance analysis are described in the following sections.

4.1 Fair ECN (FECN)

Studies have shown that the TCP ability to share a bottleneck fairly and efficiently decreases as the number of flows increase (refer Section 1.2). Several researchers observed and simulated the biased effect in TCP for connections with different RTTs. In the last chapter, the simulation results by comparing the performance of RED, "ECN over RED", DRED and "ECN over DRED" shows that ECN does provide lower retransmissions and better goodput but adds to unfairness for heterogeneous connections (TCP connections with different RTTs) (refer Section 3.4.5). These results suggest that if congestion control is to handle Web traffic consisting of thousands of concurrent connections with some degree of fairness then further enhancements to ECN are needed.

Based on the simulations results of the previous chapter, this chapter presents a fair version of ECN, "FECN over RED", that can maintain the better goodput achieve by "ECN over RED" and provide better fairness by properly adjusting the relevant ECN

parameters. Rather than treating all connections with the same max_p as in "ECN over RED", "FECN over RED" computes max_p so that connections with higher RTTs can have higher chance to get a proper share of bandwidth when competing with lower RTT connections.

4.1.1 FECN over RED Algorithm

Just like standard ECN, FECN is used together with TCP congestion control mechanism like slow-start and congestion avoidance. When an acknowledgement is not marked, the source follows the existing TCP algorithms to send data and increase the congestion window. Upon the receipt of ACK packet with Congestion Notification (CE bit set to 1), the source halves its congestion window and reduces the *ssthresh*. In the case of packet loss, the source follows the TCP algorithm to reduce the window and retransmit the lost packet.

Recall that "ECN over RED" algorithm (Section 3.1.1) detects congestion by measuring the traffic load in terms of average queue size q_{avg} using an exponentially weighted moving average filter. "FECN over RED" used the same technique for average queue size and also requires the same min_{th} and max_{th} parameters.

FECN assumes that the IP header has enough reserved space to record the round trip time of each flow. Before a source sends out a new packet, the round trip time of the last ACKed outgoing packet is added into the IP header of this new packet. FECN router computes max_p:

$$max_p = default_max_p * \frac{avg_RTT}{RTT}$$

RTT in the max_p formula is calculated by the TCP source and is in the IP Header of the incoming packet. FECN requires $default_max_p$, avg_RTT parameters. Parameter $default_max_p$ is same as max_p of RED algorithm. avg_RTT is the average value of RTT for the network connections and depend upon the network configuration. FECN algorithm computes max_p, which is high for connections with lower RTT and low for

connections with higher RTTs. So with the above formula, the dropping probability is high for connections with lower RTT and low for connections with higher RTT. Remember that TCP is biased against connections with higher RTTs. With the above formula, connections with higher RTTs can have higher chance to get a proper share of bandwidth when competing with lower RTT connections.

Following conditions define the "FECN over RED" (same as "ECN over RED") algorithm upon arrival of the packet:

- If $q_{avg} < min_{th}$, the router always queues the packet
- If $q_{avg} \geq max_{th}$, the router always drops the packet
- If $min_{th} \leq q_{avg} < max_{th}$, the router marks the packet for congestion with probability proportional to the average queue length.

The marking probability P_a is computed as (using the same algorithm as for "ECN over RED"):

$$P_a = \frac{P_b}{1 - count * P_b}$$

where $P_b = max_p * \frac{q_{avg} - min_{th}}{max_{th} - min_{th}}$

and *count* is a variable that keeps track of the number of packets that have been forwarded since the last marked packet.

Note that as the value of max_p depends upon the RTT, the marking probability of "FECN over RED" also depends upon the RTT. As mentioned above, max_p with this algorithm ("FECN over RED") is high for connections with lower RTT and low for connections with higher RTTs. So the marking probability P_a is also high for connections with lower RTT and low for connections with higher RTTs.

Table 4-1 "FECN over RED" Algorithm Parameters

Parameter Name	Parameter Symbol	Parameter Function	Recommended Value
Buffer Size	B	Buffer Size in router	1 or 2 times bandwidth-delay product
Maximum Threshold	max_{th}	Queue size control threshold	*0.6 B*
Minimum Threshold	min_{th}	Queue size control threshold	*0.2 B*
Maximum Drop Probability	$default_max_p$	The default drop probability when $min_{th} \le q_{avg} < max_{th}$,	0.1
Queue Weight	w_q	Filter Gain	0.002
Average RTT	*avg_RTT*	Average RTT of the Network	-

Control parameters for "FECN over RED" algorithm are listed in Table 4-1. All the parameters except *avg_RTT* are same as that for the RED algorithm and are repeated here for completeness and convenience. The value for this *avg_RTT* parameter depends upon the network configuration and needs to be calculated or estimated for “FECN over RED” algorithm.

4.1.2 Simulation Model

OPNET Network Model described in Section 2.4.1 is used for the performance evaluation of the ECN over AQM protocols listed above.

The “ECN over RED” implementation described in Section 2.4 is modified for “FECN over RED” protocol. The source process model, described in Section 2.4.2.1, is enhanced for “FECN over RED” as:

- Variable RTT is added into the packet header of TCP Reno. Before a source sends out a new packet, the round-trip time of this connection is put into the IP header of this new outgoing packet.
- Just like standard ECN, FECN is used together with TCP congestion control mechanism like slow-start and congestion avoidance. When an acknowledgement is

not marked, the source follows the existing TCP algorithms to send data and increase the congestion window.

- Upon the receipt of ACK packet with Congestion Notification (CE bit set to 1), the source halves its congestion window and reduces the *ssthresh*. In the case of packet loss, the source follows the TCP algorithm to reduce the window and retransmit the lost packet.

The router process model, described in Section 2.4.2.2, is enhanced for "FECN over RED" as:

- If $q_{avg} \geq max_{th}$, the router always drops the packet just as standard ECN.
- If $min_{th} \leq q_{avg} < max_{th}$
  - Get the RTT contained in the incoming packet to calculate the maximum marking probability:

$$max_p = default_max_p * \frac{avg_RTT}{RTT}$$

  - Calculate the marking probability with the new max_p and mark the packet with this new marking probability.

Table 4-2 Parameter Values for "FECN over RED" algorithm

Algorithm	Parameters	Values
FECN over RED	Minimum Threshold (min_{th})	118 packets
	Maximum Threshold (max_{th})	352 packets
	Maximum Drop Probability (max_p)	0.1
	Queue Weight (w_q)	0.002
	Buffer Size (B)	586 packets
	Average RTT (avg_RTT)	100 msec

The parameter values of the "FECN over RED" algorithm are shown in Table 4-2. The parameter values are derived using the recommended values from the original papers [FlJa93] (Section 4.1.2). Note that the parameter analysis of RED algorithm has been done in lot of papers [FlJa93, Floy97, FiBo00]; we do not focus on the parameter

sensitive analysis of these algorithms in this thesis. Since the impact of these parameters on "FECN over RED is same as that that on its base algorithm (RED), it is reasonable to use the recommended values from the original papers.

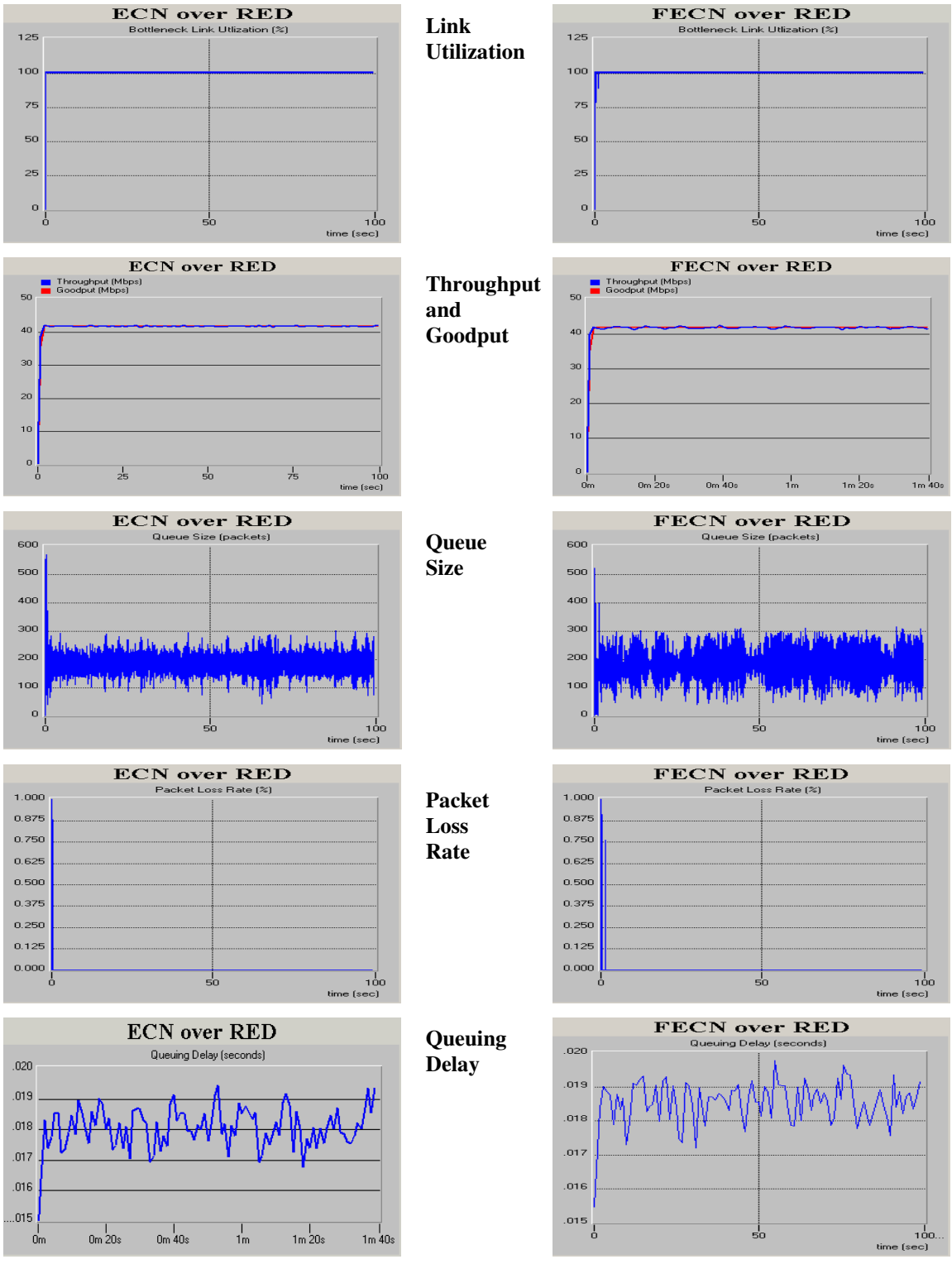
4.1.3 Simulation Scenarios

To evaluate the performance of "FECN over RED", a series of simulations with OPNET Modeler are run and performance metrics are recorded. We run the simulation on PIII-500 NT workstation. It takes around 25 minutes to run a simulation for 100 simulation seconds. Depending upon the simulation scenario the configurable parameters for the OPNET model (network, node and process model) described above are set at the simulation run time. Simulation is run for 100 seconds for each scenario to gather the key performance data.

Table 4-3 Simulation Scenarios for "FECN over RED"

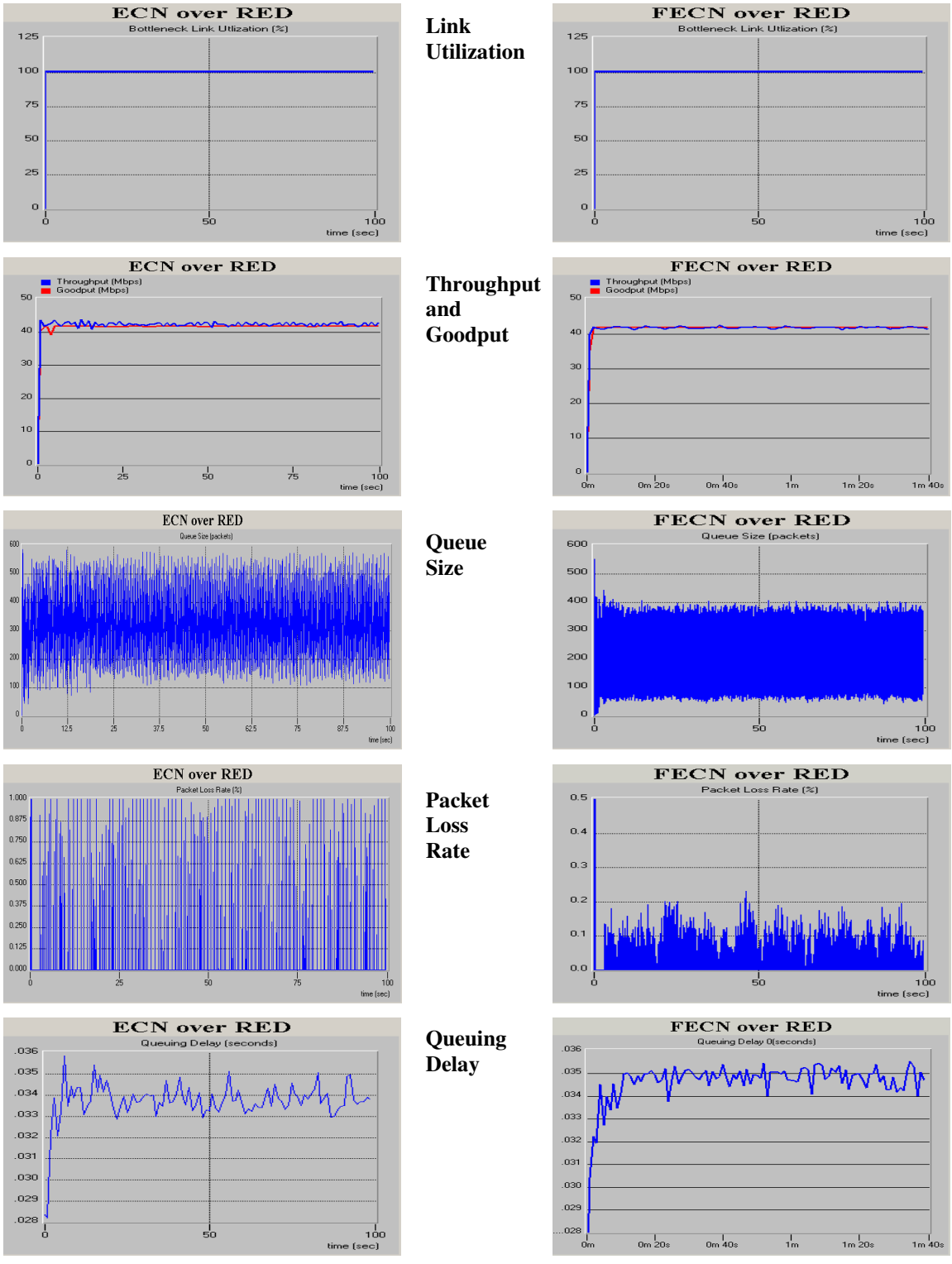
	Protocol	Network Configuration	Sources	Conn-ections	RTT (msec)
Scenario 1a	MFFECN over RED	Heterogeneous	100	100	100
Scenario 1b	MFFECN over RED	Heterogeneous	300	100	60
				100	100
				100	140
Scenario 1c	MFFECN over RED	Heterogeneous	500	100	60
				100	80
				100	100
				100	120
				100	140
Scenario 1d	MFFECN over RED	Heterogeneous	1000	100	60
				100	70
				100	80
				100	90
				100	100
				100	110
				100	120
				100	130
				100	140
				100	150

Simulation scenarios for evaluating the performance of various ECN over AQM protocols are listed in Table 4-3.



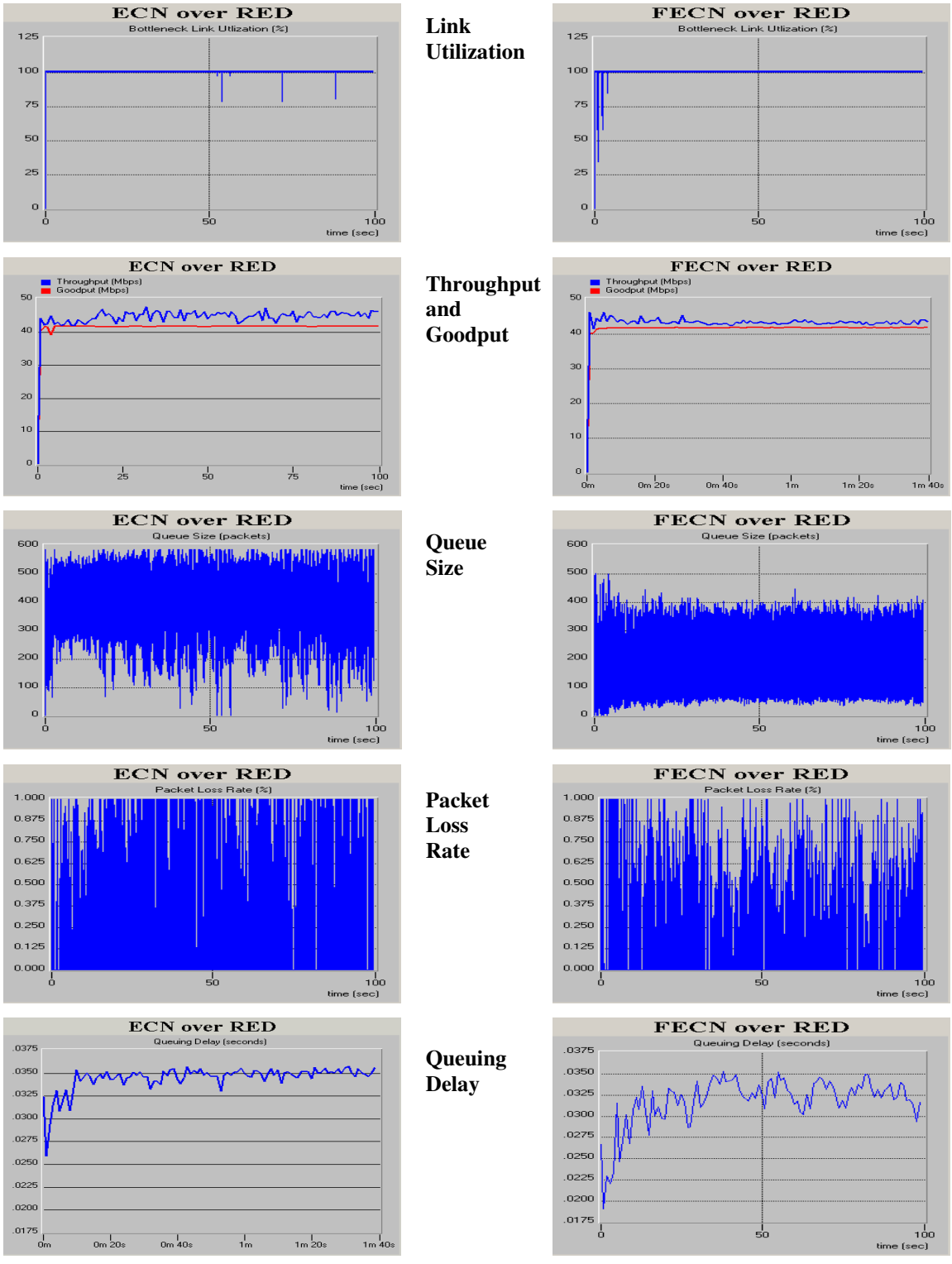
(a) ECN over RED **(b) FECN over RED**

Figure 4-1 "ECN over RED" and "FECN over RED" Performance Characteristics (100 Sources)



(a) ECN over RED **(b) FECN over RED**

Figure 4-2 "ECN over RED" and "FECN over RED" Performance Characteristics (500 Sources)



(a) ECN over RED **(b) FECN over RED**

Figure 4-3 “ECN over RED” and "FECN over RED" Performance Characteristics (1000 Sources)

4.1.4 Performance Analysis

Figure 4-1 Figure 4-2 Figure 4-3 show the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for "ECN over RED" and "FECN over RED" for 100, 500 and 1000 sources respectively.

Bottleneck link should be optimally utilized for better network performance. Recall that "ENC over RED" provided a better utilization than RED (Section 3.4.1) for all three simulation scenarios i.e. 100, 500 and 1000 sources. These graphs show that "FECN over RED" maintains that advantage, as the Bottleneck link utilization with "FECN over RED" is comparable to "ECN over RED" for all three scenarios.

Difference between throughput and goodput represents the number of retransmit packets. With RED the number of retransmit packets (i.e. difference between throughput and goodput) increase with the increase in the number of sources. Recall that this rate of increase is much lower with "ECN over RED" (Section 3.4.1). Graphs show that, for 100 sources and 500 sources simulation scenarios, the number of retransmit packets is same for both "ECN over RED" and "FECN over RED" algorithms. However, for 1000 sources simulation scenario, the difference between throughput and goodput (i.e. retransmitted packets) is lower for "FECN over RED" than "ECN over RED". So in this respect "FECN over RED" performs even better than "ECN over RED". Also note that the Goodput i.e. the effective data rate of "ECN over RED" and "FECN over RED" is same but better then RED.

Queue size of "FECN over RED" depends upon the number of sources i.e. active queue connections, similar to RED and "ECN over RED". Graphs show that for our simulation scenarios, queue overflows for "FECN over RED" are not as frequent as with RED and "ECN over RED". However, queue size is unstable, which is due to the design principle of "RED" inherited by "FECN over RED", and can lead to queue overflows and oscillation behavior in TCP sources. At higher loads, the errors or oscillations in the system can push the operating queue size beyond the maximum buffer size, resulting in packet drops. Queue size for "FECN over RED" is lower than "ECN over RED" especially for scenarios with higher number of connections (1000 sources), but still higher than for the "RED" protocol. This is because of the extra packets with are

marked and queued instead of dropped, due to congestion, with “FECN over RED” protocol.

Packet drop rate with “FECN over RED” increases with the increase in number of sources, similar to RED and “ECN over RED”. Packet drop rate is better than RED and same as “ECN over RED”, for lower number of sources. Recall that Packet Drop Rate with “ECN over RED” increases significantly with the increase in the number of sources i.e. network traffic (Section 3.4.1). This is because of the higher queue size with “ECN over RED’ protocol, which often approaches the maximum buffer size, resulting in packet drops. Graphs show that for 100 and 500 sources scenarios, the packet drop rate is same for both “ECN over RED” and “FECN over RED” protocol. However, for 1000 sources scenario, the Packet Drop Rate is lower for “FECN over RED” compared to “ECN over RED”.

Queuing delay with “FECN over RED”, “ECN over RED” and RED protocol increases with the increase in the number of sources i.e. network traffic. Recall that with “ECN over RED” the delay is higher than “RED” for all the three scenarios, which is because of the higher queue size resulting in longer time for the packets in the router queue waiting for processing. Queuing delay for “FECN over RED” is same as that for “ECN over RED” with smaller number of sources (100 sources scenario, Figure 4-1) but lower for higher number of sources (1000 sources scenario, Figure 4-2).

Recall that the objective of “FECN over RED” algorithm is to provide a better fairness in heterogeneous networks, i.e. when connections with different RTTs are competing for network resources, while maintaining or exceeding the other performance characteristics (bottleneck link utilization, throughput, goodput, queue size, packet loss rate, queuing delay) of “ECN over RED” algorithm. Fairness evaluation of “FECN over RED” is done in the next section. However, the graphs (Figure 4-1, Figure 4-2 and Figure 4-3) and above discussions show that the performance results (other than fairness) for “FECN over RED” are quite comparable to “ECN over RED” for all three simulation scenarios i.e. 100 sources, 500 sources and 1000 sources. Performance with higher traffic load (1000 sources scenario) is even better (lower number of retransmit packets, lower queue size, lower packet drop rate and lower queuing delay) for “FECN over RED” compared to “ECN over RED”.

Jain's Fairness Index

Jain's fairness index postulates that the network is a multi-user system, and drives the metrics to see how fairly each user is treated. Refer to Section 2.3.7.1 for detailed description of this algorithm. Following table shows the Jain's Fairness Index for RED, "ECN over RED" and "FECN over RED"

Table 4-4 Jain's Fairness Index: RED, "ECN over RED" and "FECN over RED"

	RED	**"ECN over RED"**	**"FECN over RED"**
3 Subnets (300 Sources) Subnet 0 RTT: 60 msec. Subnet 1 RTT: 100 msec Subnet 2 RTT: 140 msec	0.9364	0.8688	0.9989
5 Subnets (500 Source) Subnet 0 RTT: 60 msec Subnet 1 RTT: 80 msec Subnet 2 RTT: 100 msec Subnet 3 RTT: 120 msec Subnet 4 RTT: 140 msec	0.9124	0.8838	0.9931

For 3-subnets scenario with different RTTs (60 msec, 100msec and 140 msec), Jain's Fairness Index is 0.9364 for RED, 0.8477 for "ECN over RED", and 0.9989 for "FECN over RED". For 5-subnets scenario with different RTTs (60 msec, 80 msec, 100 msec, 120 msec and 140 msec), Jain's Fairness Index is 0.9124 for RED, 0.8959 for "ECN over RED" is 0.89, and 0.9931 for "FECN over RED".

Since the perfect fairness has a Jain's fairness index of 1, it's clear that "FECN over RED" provides a significantly better fairness than RED and "ECN over RED" for both 3-subnets and 5-subnets scenario. These numbers show that the improvement in fairness is about 15% for 3-subnets and 10% for 5-subnets. The 3-subnets scenario indicates the effect of large changes in RTT (40 msec) and the 5-subnets scenario indicates the effect of small changes in RTT (20 msec). Jain's Fairness Index is almost same for both the scenarios with "FECN over RED" protocol.

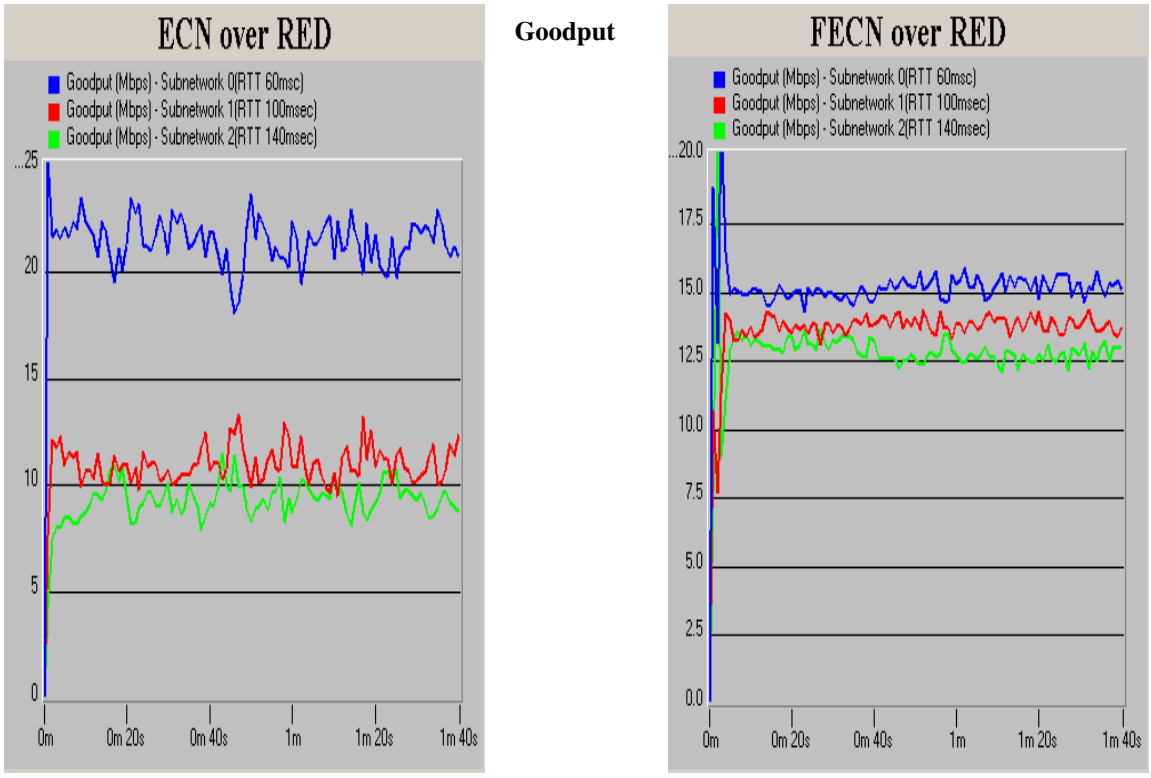


Figure 4-4 "ECN over RED" and "FECN over RED" Goodput (300 Sources)

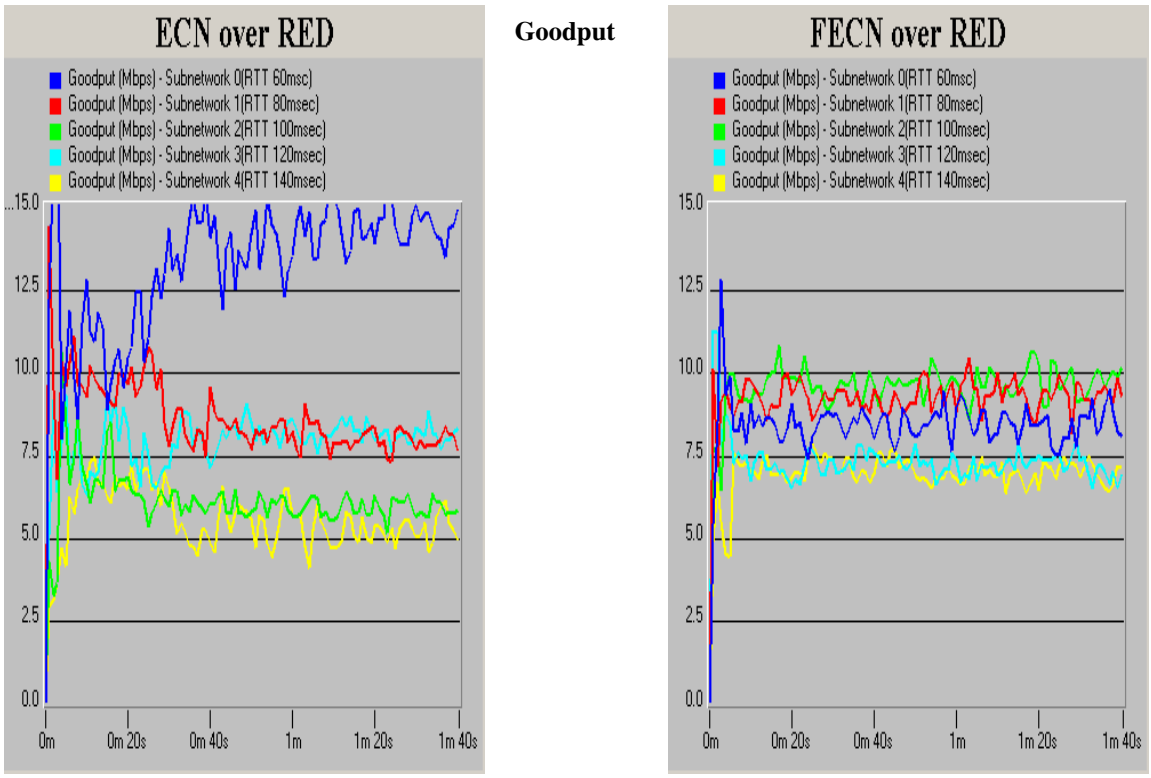


Figure 4-5 "ECN over RED" and "FECN over RED" Goodput (500 Sources)

Visual Max-Min Fairness

Figure 4-4 provides a visual sense of max-min fairness via the difference between goodput for the three subnets with different RTTs (60msec, 100msec and 140msec) for “ECN over RED” and “FECN over RED”. These graphs clearly show that, for “ECN over RED”, the goodput of an individual connection depends largely on its round-trip time and there is a bias towards shorter connections when the connections have different RTTs. In these graphs the average goodput of the sources in Subnet-0 (with lowest RTT - 60 msec) is highest, followed by that of Subnet-1 (RTT – 100msec) and is lowest for Subnet-2 (RTT - 140 msec). The graph for “FECN over RED” show that this bias towards connections with lower RTTs is much smaller for “FECN over RED”. So “FECN over RED” is fairer than RED and “ECN over RED". This observation confirms the Jain’s Fairness Index calculations.

Figure 4-5 provides a visual sense of max-min fairness via the difference between goodput for five subnets with different RTTs (60msec, 80msec, 100msec, 120msec, 140msec), for “ECN over RED” and “FECN over RED”. This scenario shows the impact on goodput with small changes in RTT values (20 msec) of completing connections, compared to the large changes (40 msec) discussed above (Figure 4-4). The graph for “ECN over RED” clearly shows the significant impact on Goodput for the small changes in RTT and the bias towards shorter connections (lower RTT). “FECN over RED” also improves the fairness significantly for this scenario. It is clear from these graphs that the goodput is almost equal for all the subnets and confirms the Jain’s Fairness Index calculations.

It is clear from the above discussion that “FECN over RED” improves the fairness by restricting the lower RTT connections with aggressive max_p and encourages the higher RTT connections with a conservative max_p. With this, "FECN over RED" lets the group of connections with higher RTTs get a higher share of bandwidth at bottleneck link than "ECN over RED". Figure 4-4 and Figure 4-5 show that the share of goodput for the higher RTT subnets increases while the share for the lower RTT subnets goes down resulting in fairness across the heterogeneous network.

4.2 Mark First Fair ECN (MFFECN)

Standard ECN and FECN, as described in the previous sections, use the mark-tail strategy for congestion control. With Mark-tail strategy, the packet that just arrived will be marked in the event of congestion in the network, but the packets already in the buffer will be sent without being marked. There are two major disadvantages of this technique. First, with this technique the notification of congestion to the source depends upon the queue size and so does not provide up-to-date congestion notification to the source. Second, this technique results in discrimination against new connections. Consider the time when a new connection joins the network, but the buffer of the congested router is occupied by the old flows. With the mark-tail strategy, the packet that just arrived will be marked, but the packets already in the buffer will be sent without being marked. The ACK of these unmarked signals will increase the window size of the old connections resulting in more bandwidth to these connections.

This section proposes a Mark-front strategy for ECN. This strategy marks the first unmarked packet in the front of the queue, instead of the new packet as in mark-tail, for congestion notification. Mark-front strategy can hasten the transmission of congestion notification to the source, since the marked packets will not have to wait in the router queue. Mark-front strategy will also alleviate the discrimination against new flows. This strategy can also help alleviate the discrimination against connections with higher RTTs. As we have seen in the previous chapters, connections with lower RTT receive their ACK faster, and therefore get more bottleneck bandwidth. In the case of congestion at the bottleneck, router queue has more packets from lower RTT connections than those from higher RTT connections. With the mark-tail strategy, the router will mark only the newly arriving packets and this may cause the lower RTT connections to grow even larger than the higher RTT connections resulting in unfairness. Mark-front strategy alleviates this discrimination by marking the packets already in the buffer. In this way, connections with higher RTTs can get larger bandwidth resulting in fairness. Combining this method with the FECN [Section 4.1] can result in efficient and fair congestion control for TCP/IP Networks.

4.2.1 MFFECN over RED Algorithm

MFFECN is used together with TCP congestion control mechanism (like slow-start and congestion avoidance), and the Fair ECN mechanism introduced in the last section. When an acknowledgement is not marked, the source follows the existing TCP algorithms to send data and increase the congestion window. Upon the receipt of ACK packet with Congestion Notification (CE bit set to 1), the source halves its congestion window and reduces the *ssthresh*. In the case of packet loss, the source follows the TCP algorithm to reduce the window and retransmit the lost packet.

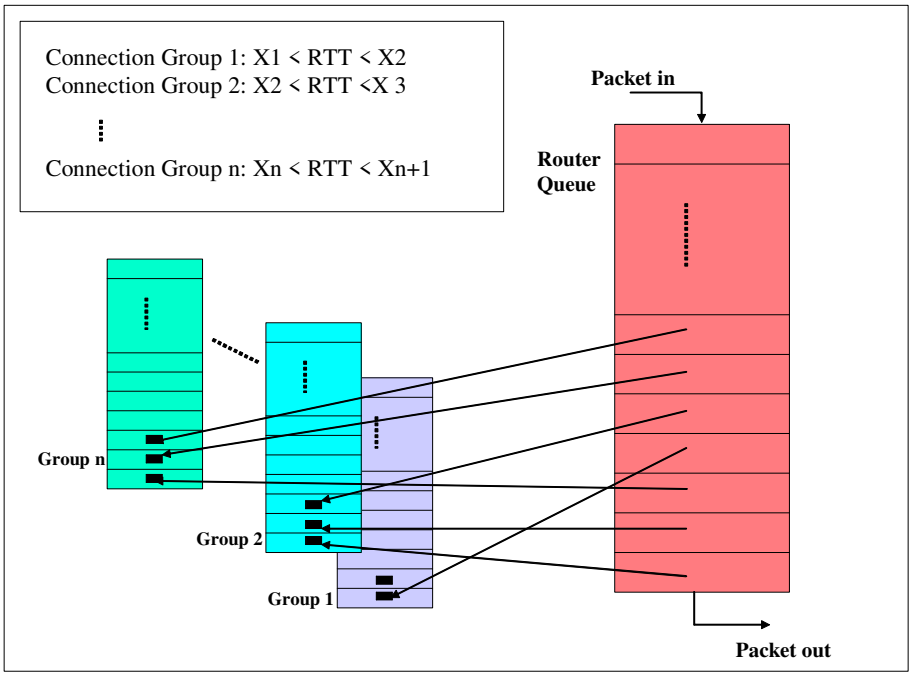


Figure 4-6 FECN Flow Group Queues and Router Queue

MFFECN used the same technique as "ECN over RED" (refer Section 3.1.1) for average queue size and also requires the same min_{th} and max_{th} parameters.

MFFECN also assumes, similar to FECN, that the IP header contains the round trip time of the last ACKed outgoing and computes max_p from RTT as:

$$max_p = default_max_p * \frac{avg_RTT}{RTT}$$

MFFECN divides all connections into various groups depending upon their RTT. The number of these groups and the RTT range for a specific group is configurable for a router and depends upon the network configuration and traffic assumptions. Each group corresponds to a virtual queue in the router storing the address of each packet in the main router queue. The relationship between the FECN router queue and connection groups is shown in Figure 4-6. Each group maintains a range of RTTs to be compared with the RTT in the arriving packet. Based on the RTT in the IP header of the arriving packet, router decides which connection group the incoming packet belongs to and on the maximum marking probability as per the formulae defined above. The number of the control groups in a network configuration depends upon the RTT range of all the network connections. Note that using too many groups can lead to high service time at the router. However, with small number of connection groups (i.e. each connection group with high RTT range) can lead to unfairness in the network.

Following conditions define the "FECN over RED" algorithm upon arrival of the packet:

- If $q_{avg} < min_{th}$, the router always queues the packet and updates the connection groups depending upon the RTT of incoming packet
- If $q_{avg} \geq max_{th}$, the router always drops the packet
- If $min_{th} \leq q_{avg} < max_{th}$, the router queues the packet, updates the connection groups, and marks the first unmarked packet of the queued packet connection group for congestion with the marking probability P_a.

  The marking probability P_a is computed as (same as that for RED):

$$P_a = \frac{P_b}{1 - count * P_b}$$

$$where\ P_b = max_p * \frac{q_{avg} - min_{th}}{max_{th} - min_{th}}$$

and *count* is a variable that keeps track of the number of packets that have been forwarded since the last marked packet.

Similar to FECN, as the value of max_p depends upon the RTT, the marking probability of "FECN over RED" also depends upon the RTT.

Table 4-5 "MFFECN over RED" Algorithm Parameters

Parameter Name	Parameter Symbol	Parameter Function	Recommended Value
Buffer Size	*B*	Buffer Size in router	1 or 2 times bandwidth-delay product
Maximum Threshold	max_{th}	Queue size control threshold	*0.6 B*
Minimum Threshold	min_{th}	Queue size control threshold	*0.2 B*
Maximum Drop Probability	$default_max_p$	The maximum drop probability when $min_{th} \le q_{avg} < max_{th}$,	0.1
Queue Weight	w_q	Filter Gain	0.002
Average RTT	*avg_RTT*	Average RTT of the Network	-
Number of Groups	*n*	Total number of connection groups	-
Group RTT	*RTT-n*	RTT range for each group	-

Control parameters for "MFFECN over RED" algorithm are listed in Table 4-5. All the parameters except the last three *(avg_RTT, n, RTT-n*) are same as that for the RED algorithm and are repeated here for completeness and convenience. The value for these parameters (*avg_RTT, n, RTT-n*) depends upon the network configuration and needs to be calculated or estimated for “MFFECN over RED” algorithm.

4.2.2 Simulation Model

TCP/IP bottleneck network configuration (Figure 2-6), with two routers and a number of subnets (with TCP sources), use for the performance evaluation is same as the one used in the last chapter.

The "FECN over RED" implementation described in Section 4.1.2 is modified to implement "MFFECN over RED" protocol. The source process model, described in Section 4.1.2 for "FECN over RED" remains the same to enable the RTT in the packet header of TCP Reno.

The router process model, described in Section 4.1.2 for "FECN over RED", is enhanced for "MFFECN over RED" as:

a. Implement the Flow Groups as described in Section 4.2.1
   - Flow groups are implemented using OPNET subqueues [Opnet]
   - Current model supports three connection groups with default RTTs as:
     - Flow Group 1: RTT: 0 to 80 msec
     - Flow Group 2: RTT: 81 to 120 msec
     - Flow Group 3: RTT: 121 and above

b. When a new packet comes to the router, MFFECN Router checks:
   - If $q_{avg} \geq max_{th}$, the router always drops the packet just as standard ECN.
   - If $min_{th} \leq q_{avg} < max_{th}$
     - Get the RTT contained in the incoming packet to calculate the maximum marking probability P_a as described in Section 4.1.1
     - Use the RTT of the incoming packet to decide which connection group this packet belongs and insert the packet in the corresponding subqueue.
     - Mark the first unmarked packet in that subqueue with marking probability P_a, as calculated above.
   - If $q_{avg} \leq min_{th}$
     - Use the RTT of the incoming packet to decide which connection group this packet belongs and insert the packet in the corresponding subqueue.

Table 4-6 Parameter Values for "MFFECN over RED" algorithm

Algorithm	Parameters	Values
MFFECN over RED	Minimum Threshold (min_{th})	118 packets
	Maximum Threshold (max_{th})	352 packets
	Maximum Drop Probability (max_p)	0.1
	Queue Weight (w_q)	0.002
	Buffer Size (*B*)	586 packets
	Average RTT (*avg_RTT*)	100 msec
	Number of Groups (*n*)	3
	RTT of Group 1 (*RTT-1*)	0 - 80 msec
	RTT of Group 2 (*RTT-2*)	81 - 120 msec
	RTT of Group 3 (*RTT-3*)	> 120 msec

The parameter values of the "MFFECN over RED" algorithm are shown in Table 4-6. The parameter values are derived using the recommended values from the original papers [FlJa93] (Section 4.2.2). Note that the parameter analysis of RED algorithm has been done in lot of papers [Floy97, FlJa93, FiBo00]; we do not focus on the parameter sensitive analysis of these algorithms in this thesis. Since the impact of these parameters on “MFFECN over RED” is same as that that on the base algorithm (RED), it is reasonable to use the recommended values from the original papers.

4.2.3 Simulation Scenarios

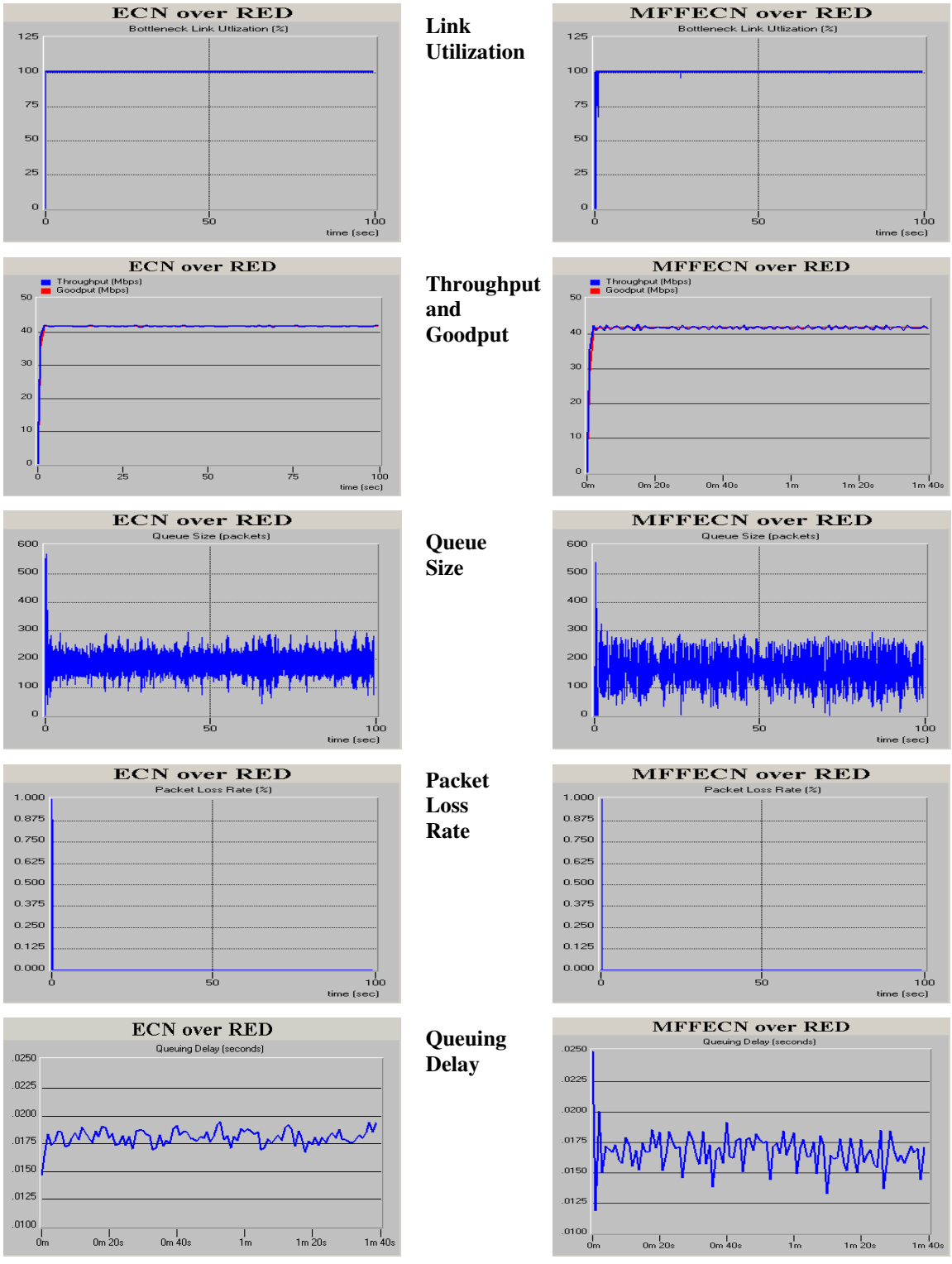
To evaluate the performance of “MFFECN over RED”, a series of simulations with OPNET Modeler are run and performance metrics are recorded. We run the simulation on PIII-500 NT workstation. It takes around 30 minutes to run a simulation for 100 simulation seconds. Depending upon the simulation scenario the configurable parameters for the OPNET model (network, node and process model) described above are set at the simulation run time. Simulation is run for 100 seconds for each scenario to gather the key performance data.

Table 4-7 Simulation Scenarios for "MFFECN over RED"

	Protocol	Network Configuration	Sources	Conn-ections	RTT (msec)
Scenario 1a	MFFECN over RED	Heterogeneous	100	100	100
Scenario 1b	MFFECN over RED	Heterogeneous	300	100	60
				100	100
				100	140
Scenario 1c	MFFECN over RED	Heterogeneous	500	100	60
				100	80
				100	100
				100	120
				100	140
Scenario 1d	MFFECN over RED	Heterogeneous	1000	100	60
				100	70
				100	80
				100	90
				100	100
				100	110
				100	120
				100	130
				100	140
				100	150

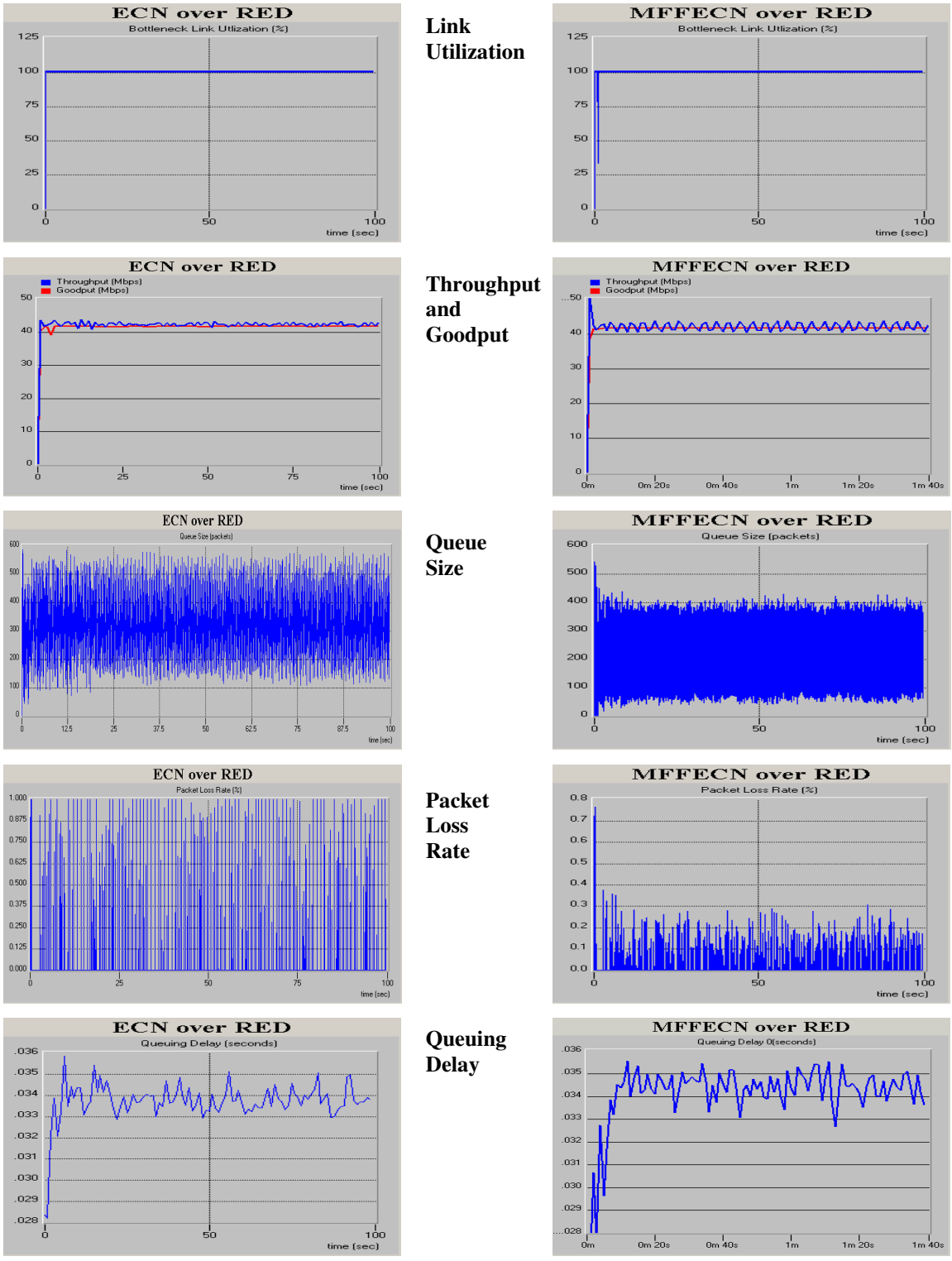
Simulation scenarios for evaluating the performance of various ECN over AQM protocols are listed in Table 4-7. These simulation scenarios are designed to compare the performance of “FECN over RED” protocol with “ECN over RED” protocol results.

The objective of “MFFECN over RED” is to provide better performance (throughput, goodput, packet loss etc.) while maintaining the fairness advantage achieved by “FECN over RED”. The simulation scenarios listed above are designed with this objective in mind. Simulation is run for 100 seconds with “MFFECN over RED” algorithm and performance characteristics (bottleneck link utilization, throughput, goodput, queue size, packet loss rate and queuing delay) are collected to compare with the performance characteristics for “ECN over RED”. Fairness for “MFFECN over RED” is measured using the Jain’s Fairness Index and visual max-min technique in a heterogeneous network, i.e. connections with different RTTs competing for network resources (Scenario 2a and 2b from Table 4-7).



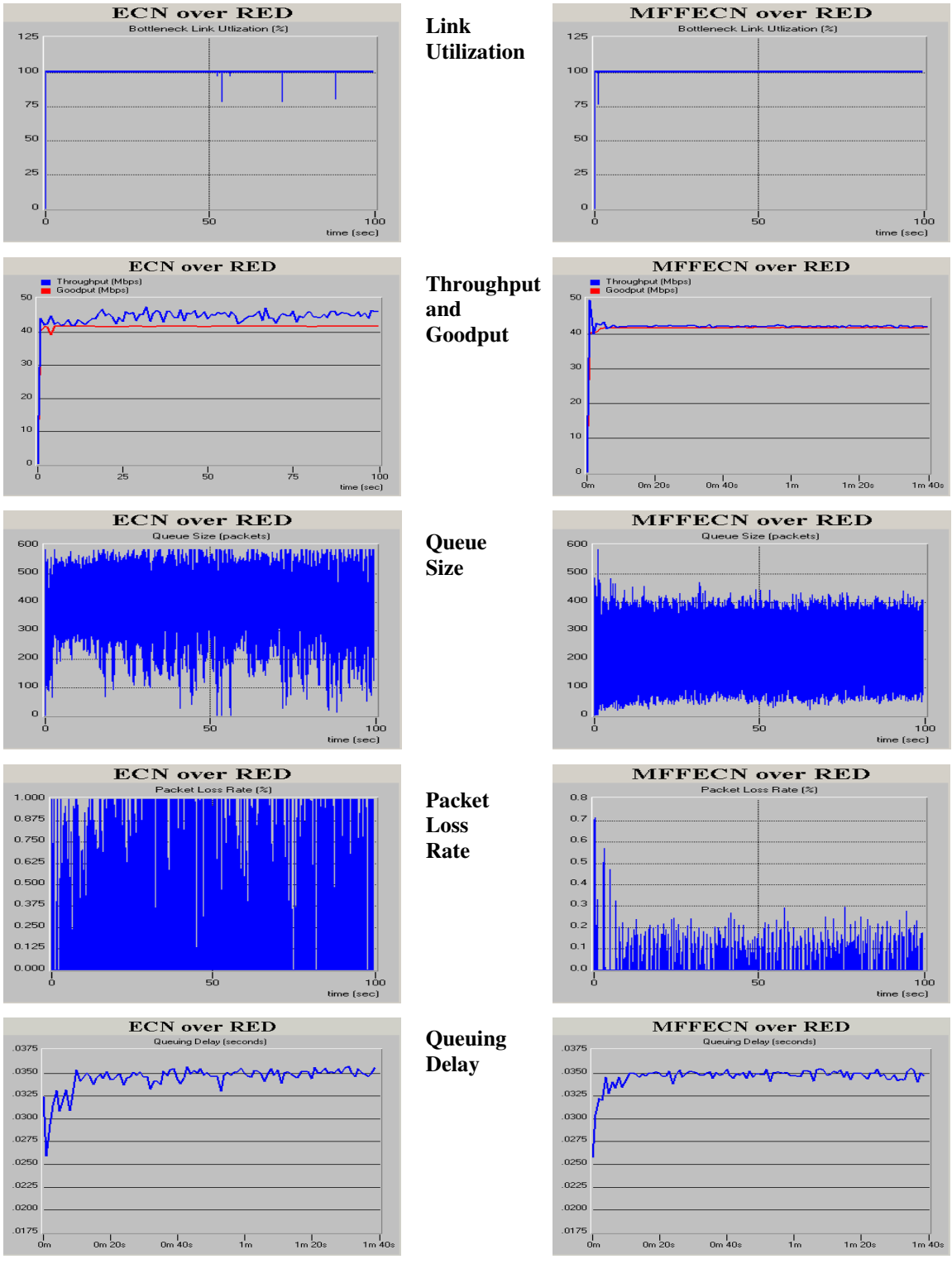
(a) ECN over RED **(b) MFFECN over RED**

Figure 4-7 "ECN over RED" and "MFFECN over RED" Performance Characteristics (100 Sources)



(a) ECN over RED **(b) MFFECN over RED**

Figure 4-8 "ECN over RED" and "MFFECN over RED" Performance Characteristics (500 Sources)



(a) ECN over RED **(b) MFFECN over RED**

Figure 4-9 "ECN over RED" and "FECN over RED" Performance Characteristics (1000 Sources)

4.2.4 Performance Analysis

Figure 4-7, Figure 4-8 and Figure 4-9 show the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for "MFFECN over RED" and "ECN over RED" for 100, 500 and 1000 sources respectively.

Bottleneck link should be optimally utilized for better network performance. Recall that "ENC over RED" provided a better utilization than RED (Section 3.4.1) for all three simulation scenarios i.e. 100, 500 and 1000 sources. These graphs show that "MFFECN over RED" maintains that advantage, as the bottleneck link utilization with "MFFECN over RED" is comparable to "ECN over RED" for all three scenarios.

The number of retransmit packets, difference between throughput and goodput, increase with the increase in the number of sources for RED. Recall that this rate of increase is much lower with "ECN over RED" (Section 3.4.1). Graphs show that for 100 sources and 500 sources simulation scenarios, the difference between throughput and goodput (i.e. retransmitted packets) is same for "MFFECN over RED" and "ECN over RED". However, for 1000 sources simulation scenario, the number of retransmit packets is lower for "MFFECN over RED" compared to "ECN over RED". So in this respect "MFFECN over RED" performs better than "ECN over RED". Also note that the Goodput, i.e. the effective data rate, for "ECN over RED", "FECN over RED" and "MFFECN over RED" are same but better then RED.

Queue size of "MFFECN over RED" depends upon the number of sources i.e. active queue connections, same as that of RED, "ECN over RED" and "FECN over RED". Graphs show that for our simulation scenarios queue overflows for "MFFECN over RED" are not as frequent as with RED and "ECN over RED", especially with higher traffic load (500 sources and 1000 sources scenarios). However, queue size is unstable, which is due to the design principle of "RED" inherited by "MFFECN over RED", and can lead to queue overflows and oscillation behavior in TCP sources. At higher loads, the errors or oscillations in the system can push the operating queue size beyond the maximum buffer size, resulting in packet drops. Queue size for "MFFECN over RED" is lower than "ECN over RED", especially for scenarios with higher number of connections, but still higher than the queue size for the "RED" protocol. This is

because of the extra packets that are marked and queued instead of dropped, due to congestion, with “FECN over RED” protocol.

Packet drop rate with “MFFECN over RED” increases with the increase in number of sources, similar to RED, “ECN over RED” and “FECN over RED”. Recall that packet drop rate with “ECN over RED” increases significantly with the increase in the number of sources i.e. network traffic (Section 3.4.1). This is because of the higher queue size with “ECN over RED’ protocol, which often approaches the maximum buffer size, resulting in packet drops. Packet drop rate with “MFFECN over RED” is lowest compared to all other protocols i.e. RED, “ECN over RED” and “FECN over RED”. This is especially clear from the graphs with higher number of connections (500 sources and 1000 sources scenarios).

Queuing delay with “MFFECN over RED” protocol increases with the increase in the number of sources i.e. traffic. Recall that with “ECN over RED” the delay is higher than “RED” for all the three scenarios, which is because of the higher queue size resulting in longer time for the packets in the router queue waiting for processing. Queuing delay for “MFFECN over RED” is same as that for “ECN over RED” with smaller number of sources (Scenario with 100 Connections Figure 4-9) but lower for higher number of sources (Scenario with 1000 connections Figure 4-11).

Recall that the objective of “MFFECN over RED” algorithm is to provide better performance (throughput, goodput, packet loss etc.) while maintaining the fairness advantage achieved by “FECN over RED”. Fairness evaluation of “MFFECN over RED” is done in the next section. However, the graphs (Figure 4-7, Figure 4-8 and Figure 4-9) and above discussions show that the performance results (other than fairness) for “MFFECN over RED” are quite comparable to “ECN over RED” for low traffic loads (100 sources scenario). Performance with higher traffic load (500 sources and 1000 sources scenarios) is better (lower number of retransmit packets, lower queue size, lower packet drop rate) for “MFFECN over RED” compared to “ECN over RED”.

Jain's Fairness Index

Jain's fairness index postulates that the network is a multi-user system, and drives the metrics to see how fairly each user is treated. Refer to Section 2.3.7.1 for detailed description of this algorithm.

T**a**ble 4-8 shows the Jain's Fairness Index for RED, "ECN over RED", "FECN over RED" and "MFFECN over RED".

Table 4-8 Jain's Fairness Index: RED, "ECN over RED" and "FECN over RED"

	RED	**"ECN over RED"**	**"FECN over RED"**	**"MFFECN over RED"**
3 Subnets (300 Sources) Subnet 0 RTT: 60 msec. Subnet 1 RTT: 100 msec Subnet 2 RTT: 140 msec	0.9364	0.8688	0.9988	0.9783
5 Subnets (500 Source) Subnet 0 RTT: 60 msec Subnet 1 RTT: 80 msec Subnet 2 RTT: 100 msec Subnet 3 RTT: 120 msec Subnet 4 RTT: 140 msec	0.9124	0.8838	0.9931	0.9961

For 3 subnets scenario with different RTTs (60 msec, 100msec and 140 msec), Jain's Fairness Index is 0.9364 for RED, 0.8477 for "FECN over RED", 0.9988 for "FECN over RED" and 0.9783 for "MFFECN over RED". For 5 subnets scenario (500 connections) with different RTTs (60 msec, 80 msec, 100 msec, 120 msec and 140 msec), Jain's Fairness Index is 0.9124 for RED, 0.8959 for "ECN over RED", 0.9931 for "FECN over RED", and is 0.9961 for "MFFECN over RED".

Since the perfect fairness has a Jain's fairness index of 1, it's clear that "MFFECN over RED" provides a significantly better fairness than RED and "ECN over RED" for both 3 subnets and 5 subnets scenario. However, the fairness index for "MFFECN over RED" is lower than "FECN over RED" for 300 connections scenario. The number shows that the improvement in fairness over "ECN over RED" is about 15% for 3 subnets and 10% for 5 subnets.

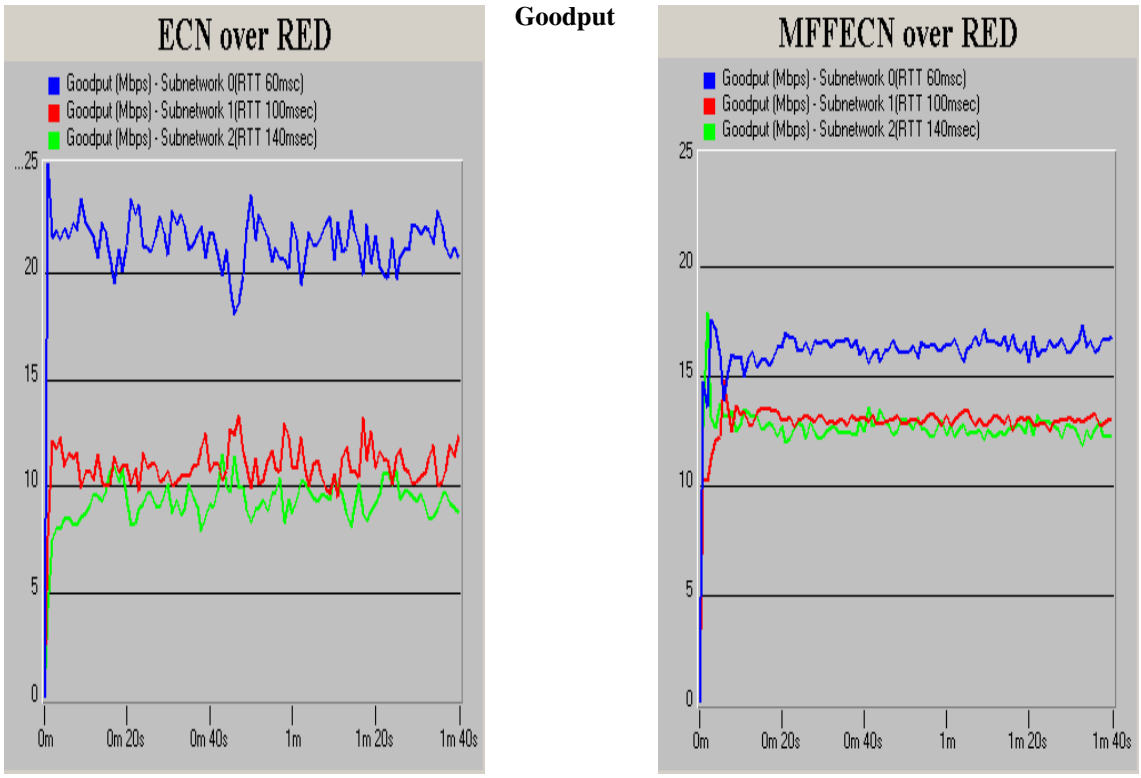


Figure 4-10 "ECN over RED" and "MFFECN over RED" Goodput (300 Sources)

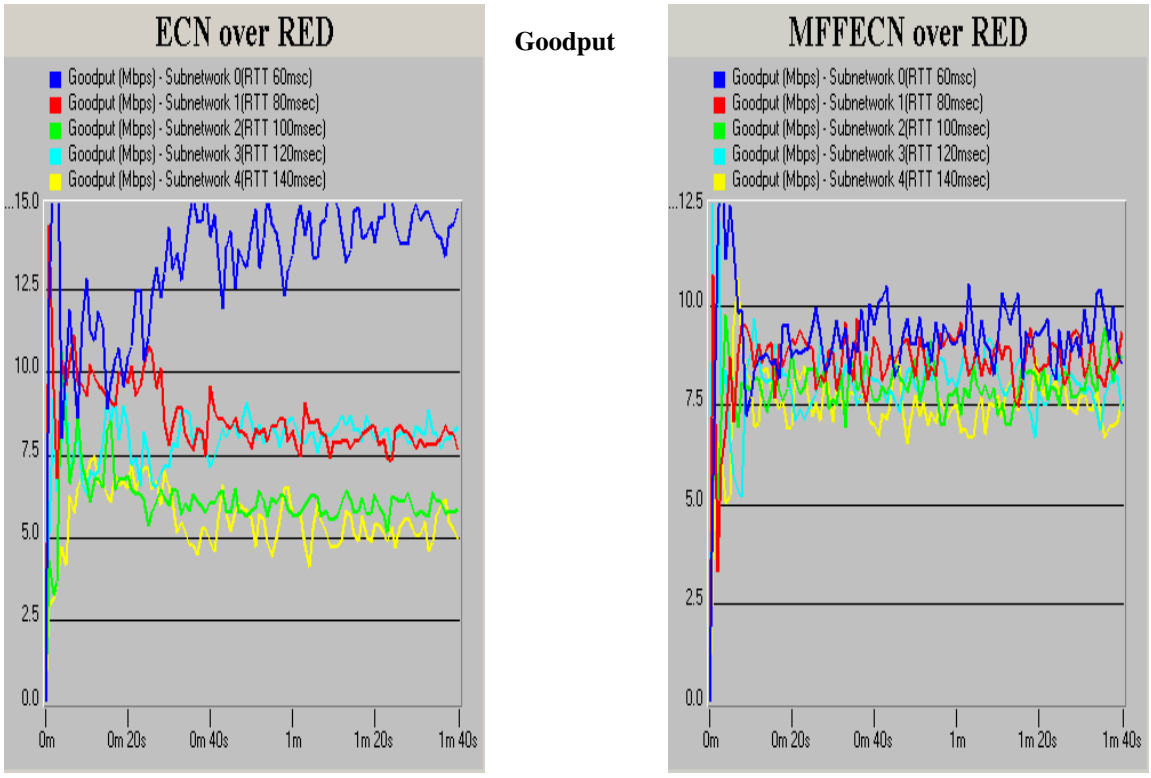


Figure 4-11 "ECN over RED" and "MFFECN over RED" Goodput (500 Sources)

Visual Max-Min Fairness

Figure 4-10 provides a visual sense of max-min fairness via the difference between goodput for the three subnets with different RTTs (60msec, 100msec and 140msec) for "ECN over RED" and "MFFECN over RED". These graphs clearly show that, for "ECN over RED", the goodput of an individual connection depends largely on its round-trip time and there is a bias towards shorter connections when the connections have different RTTs. In this graph the average goodput of the sources in Subnet-0 (with lowest RTT - 60 msec) is highest, followed by that of Subnet-1 (RTT – 100msec) and is lowest for Subnet-2 (RTT - 140 msec). The graph for "MFFECN over RED" show that this bias towards connections with lower RTTs is much smaller for "MFFECN over RED", and so is fairer than RED and "ECN over RED. This behavior is similar to "FECN over RED" (Section 4.1.4). This observation also confirms the Jain's Fairness Index calculations.

Figure 4-11 provides a visual sense of max-min fairness via the difference between goodput for five subnets with different RTTs (60msec, 80msec, 100msec, 120msec, 140msec), for "ECN over RED" and "MFFECN over RED". Note that the incremental increase in RTT from the lowest RTT connections is 20 msec in this scenario compared to 40 msec in the 3 subnet scenario discussed above. This scenario shows the impact on goodput with small changes in RTT values (20 msec) of completing connections, compared to the large changes (40 msec) discussed above (Figure 4-4). The graph for "ECN over RED" clearly shows the significant impact on Goodput for the small changes in RTT and the bias towards shorter connections (lower RTT). "MFFECN over RED" also improves the fairness significantly for this scenario. It is clear from these graphs that the goodput is almost equal for all the subnets and confirms the Jain's Fairness Index calculations.

It is clear from the above discussion that "MFFECN over RED" maintains the Fairness improvements, achieved by "FECN over RED", by restricting the connections with lower RTTs with aggressive max_p and encourages the connections with higher RTTs with a conservative max_p.

4.3 Concluding Remarks

This chapter presented enhancements to the basic algorithm of "ECN over RED", namely "FECN over RED" and "MFFECN over RED", followed by a series of simulations using OPNET Modeler that evaluates the behavior and performance of these algorithms.

The simulation results show (using Jain's fairness index and visual min-max fairness) that "FECN over RED" can treat each flow fairer when multiple flows with different RTTs are competing for the network resources "FECN over RED" provides fairness while maintaining the other advantages of "ECN over RED" (e.g. better goodput). The simulation results of "MFFECN over RED" show that it provides better goodput and lower delay while maintaining the fairness advantage achieved by "FECN over RED" protocol.

5 Design Guidelines

End-to-end congestion control and avoidance has become one of the most important and essential technologies for the transmission over the Internet. Network Services require the TCP/IP network robust enough to handle all kind of network traffic. Basic targets to design the network with high QoS are low delay and high goodput (effective data rate). This chapter provides the design guidelines based on out performance evaluations, for using ECN over AQM protocols in TCP/IP Networks to achieve superior performance.

In order to achieve the delay and data rate performance objectives, the first thing is that the congestion control algorithm should be able to minimize the packet loss. Lost packets need to be retransmitted, and they can cause long delay and waste the network resources. RED and DRED protocols have a higher packet drop rate because of the packet drop policy used by these protocols to indicate congestion. “ECN over RED” and “ECN over DRED” protocols can achieve the lower packet drop rate. However, at higher traffic loads (500 and 1000 sources scenarios) ECN results in frequent router queue overflows resulting in packet loss. “MFFECN over RED” speeds up the congestion notification to the TCP senders resulting in lower instances of queue overflow i.e. lower packet loss.

A stabilized router queue is very important for the TCP/IP Network performance. With a stable queue size, we can easily design the buffer size and the queue delay boundary in the network. Instability in queue size can lead to queue overflows and oscillation behavior in TCP sources. Increasing the buffer size to work around instability and queue oscillations in the system is not the viable option since this can lead to unacceptable and unnecessary higher queuing delay. RED protocol and other protocols based on RED (ECN over RED, MECN over RED, FECN over RED, MFFECN over RED) are unstable. Router queue size for all these algorithms depend upon the traffic load. However, DRED is the only protocol that is able to stabilize the queue size around the target value. This advantage is lost with “ECN over DRED” as the queue size increases with the network traffic load. With ECN protocols in the TCP/IP networks, it is important to note that the queue size is always higher because of the marked packets. This should be kept in mind while designing the router buffer size.

Fairness among competing flows in the Internet has become increasingly important with the growth of the Web. The absence of fairness could lead to one flow starving out another flow in the time of congestion. RED, "ECN over RED", DRED and "ECN over DRED" provide unfair bandwidth allocation in heterogeneous network (multiple connections with different RTTs). However, "FECN over RED" and "MFFECN over RED" can treat the traffic fairly. These protocols provide upto 15% improvement in the fairness (refer Section 4.2.4).

ECN based protocols ("ECN over RED", "ECN over DRED", "MECN over RED", "FECN over RED", "MFFECN over RED") require support from router as well as from end hosts, i.e. the end hosts TCP/IP stacks needs to be modified. With the large number of end hosts in most of the current TCP/IP Network environment (especially Internet), it is not practical to expect a network with all ECN capable hosts. To support the incremental deployment of ECN capable TCP hosts, it is necessary that a router identifies that the packet is ECN capable, and should only mark the packets that are from ECN capable hosts. Router should drop the packets from the non-ECN hosts, as per the underlying AQM algorithm (RED, DRED), to indicate congestion.

A major disadvantage or potential problem with ECN, non-compliant ECN hosts and potential loss of ECN messages in the network, should be considered when designing the TCP/IP Networks with ECN algorithms ("ECN over RED", "ECN over DRED", "FECN over RED", "MFFECN over RED"). A non-compliant ECN host could set the ECN field to indicate that it was ECN-capable, and the ignore ECN notifications. However, for a network that uses only packet drops for congestion notifications (RED, DRED), a non-compliant connection could also refrain from making appropriate window decreases in response to packet drops.

6 Conclusions and Future Work

6.1 Conclusions

This study presents the “ECN over RED” and “ECN over DRED” algorithms, followed by a series of simulations using OPNET Modeler that evaluates the behavior and performance of these algorithms.

The results show that ECN does provide better goodput, i.e. the effective data rate, for both RED and DRED. The difference between throughput and goodput, i.e. the number of retransmitted packets, are also lower for both "ECN over RED" and "ECN over DRED". Queue size of "ECN over RED" depends upon the number of sources i.e. active queue connections. This is a major drawback of RED and "ECN over RED" as the instability and oscillations in the queue size can lead to queue overflows and oscillation behavior in TCP sources. DRED is the only protocol that is able to stabilize the queue size around the target value. This advantage is lost with “ECN over DRED” as the queue size increases with the network traffic load. Queue size with both "ECN over RED" and “ECN over DRED” is higher than RED and DRED. Higher queue size results in higher queuing delay as packets stay longer in the queue waiting for processing. Higher queue size with ECN pushes the operating queue size beyond the buffer size resulting in instability in performance and sustained steady-state errors.

The simulation results from the heterogeneous network (connections with different RTTs) show the effect of RTT on TCP fairness with RED, “ECN over RED”, DRED, and “ECN over DRED” using Jain’s Fairness Index and Visual Max-Min Fairness. The results show neither of these protocols is fair when TCP connections with different RTTs are competing for the resources. These protocols show a strong bias towards connections with lower RTTs.

Based on the “ECN over RED” results, this study proposes enhancements in the form of “FECN over RED” and “MFFECN over RED”. “FECN over RED” provides an algorithm to provide higher bandwidth to connections with higher RTTs when competing with connections with lower RTTs. “MFFECN over RED” proposes the mark-front strategy instead of mark-tail strategy in standard “ECN over RED” to speed up the congestion notification to the sender.

The simulations result show that “FECN over RED” and “MFFECN over RED” can treat the traffic fairly than standard “ECN over RED. Results show that “MFFECN over RED” provides better goodput and lower delay while maintaining the fairness advantage achieved by “FECN over RED” protocol.

6.2 Future Work

The following is a list of work that can be done in the future:

1. Investigate the performance of “MFFECN over RED” in more complicated network configurations e.g. networks with multiple congested routers, and different traffic patterns e.g. short flows
2. Analyze ECN over other AQM techniques like Adaptive RED (ARED), BULE etc. As we have seen with “ECN over RED” and “ECN over DRED”, ECN introduces various new issues when implemented over the AQM techniques and these issues depend upon the particular AQM technique. So it is important to analyze them properly.
3. Explore the load adaptive technique [AwOu01b, FlGu02] to further refine the “MFFECN over RED” protocol to provide stable queue size. Queue size of "MFFECN over RED" depends upon the number of sources i.e. active queue connections, which can lead to queue overflows and oscillation behavior in TCP sources.
4. Apply ECN Enhancements proposed for RED to other AQM techniques.

7 Bibliography

[AwOu01a] J. Aweya, M. Ouellette, and D. Y. Montuno. "A Control Theoretic Approach to Active Queue Management", Published in Computer Networks Journal (Elsevier Science), Vol. 36, Issue 2-3, pp. 203-235, July 2001.

[AwOu01b] J. Aweya, M. Ouellette, D. Y. Montuno, A. Chapman. "Enhancing TCP Performance with a Load adaptive RED Mechanism". International Journal of Network Management, V.11, N.1, 2001.

[BaKa99] Prasad Bagal, Shivkumar Kalyanaraman, Bob Packer, "Comparative study of RED, ECN and TCP Rate Control," Technical Report, March 1999 http://www.ecse.rpi.edu/Homepages/shivkuma/research/papers/packeteer-final.pdf.

[BrOM94] L. Brakmo, S. O'Malley, and L. Peterson. "TCP Vegas: New techniques for congestion detection and avoidance". In *Proceedings of the SIGCOMM '94 Symposium* (Aug. 1994) pp. 24-35.

[FaFl96] K. Fall, and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP" Computer Communication Review, V. 26 N. 3, July 1996, pp. 5-21

[FeKa99] W. Feng, D. Kandlur, D. Saha, K. Shin, "Blue: A New Class of Active Queue Management Algorithms" U. Technical Report CSE-TR-387-99, Dept. of EECS, University of Michigan. April 1999.

[FiBo00] Victor Firoiu and Marty Borden, "A Study of Active Queue Management for Congestion Control", IEEE INFOCOM2000, Tel Aviv, Israel, March 2000, pp. 1435-1444.

[FlFa99] S. Floyd and K. Fall. "Promoting the Use of End-to-End Congestion Control in the Internet". IEEE/ACM Transactions on Networking, August 1999.

[FlHe99] Sally Floyd, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm. Internet Draft draft-ieft-tcpimpl-newreno-02.txt, Feb. 1999.

[FlJa92] Sally Floyd, Van Jacobson. "Traffic Phase Effects in Packet-Switched Gateways". Internetworking: Research and Experience, Vol. 3, No. 3, September 1992.

[FlJa93] Sally Floyd, Van Jacobson. "Random Early Detection Gateways for Congestion Avoidance". IEEE/ACM Transactions on Networking, V.1 N.4 August 1993, pp. 397-413.

[FlGu02] S. Floyd, R. Gummadi, and S. Shenker. "Adaptive RED: An Algorithm for Increasing the Robustness of RED". Technical Report, to appear, 2002.

[FlMa00] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky. "An Extension to the Selective Acknowledgement (SACK) Option for TCP" RFC 2883. July 2000.

[Floy91a] S. Floyd. "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic". Computer Communication Review, Vol.21, No.5, October 1991, pp. 30-47

[Floy91b] S. Floyd. “Connections with Multiple Congested Gateways in Packet-Switched Networks Part 2: Two-way Traffic”. http://www.icir.org/floyd/papers.html

[Floy94] Sally Floyd. "TCP and Explicit Congestion Notification". ACM Computer Communication review, Vol. 24, no. 5, October 1994, pp. 10-23.

[Floy97] Sally Floyd, “RED: Discussions of Setting Parameters”, November 1997. http://www.aciri.org/floyd/REDparameters.txt.

[HoMi01a] C.V. Hollot, Vishal Mishra, Don Towsley, and Wei-Bo Gong, “A Control Theoretic Analysis of RED”, Proc of IEEE INFOCOM2001, Alaska USA, April 22-26 2001, pp. 1510-1519

[Jaco88] Van Jacobson. "Congestion Avoidance and Control". In Proceedings of ACM SIGCOMM, 1988, pp. 314-329

[Jaco90] Van Jacobson. “Modified TCP Congestion Avoidance Algorithm”. Technical report, 30 April 1990. Email to the end2end-interest Mailing List, URL ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[Jain91] R. Jain. “The Art of Computer Systems Performance Analysis”. John Wiley and Sons, QA76.9.E94J32, 1991

[LiJa01] C. Liu, R. Jain, “Improving Explicit Congestion Notification with Mark-Front Strategy”, Computer Networks, vol.35, no. 2-3, pp.185-201, January 2001

[MaMa96a] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow. “TCP Selective Acknowledgment Options”. RFC 2018. 1996.

[MaMa96b] M. Mathis, J. Mahdavi, “Forward Acknowledgement Refining TCP Congestion Control”, SIGCOMM Symposium on Communication Architectures and Protocols, Aug. 1996.

[Mank97] Allison Mankin. "Random Drop Congestion Control". In Proceeding of ACM, SIGCOMM, 1997.

[MaSe97] Matthew Mathis, Jefferey Semke, Jamshid Mahdavi, and Teunis Ott, “The Microscopic Behavior of the TCP Congestion Control”, ACM Computer Communication Review, col. 27, No.3, July 1997.

[Morr97] Robert Morris, “TCP Behavior with Many Flows”, IEEE International Conference on Network Protocols, October 1997, Atlanta, Georgia, pages 205-211[Morr99a] Robert Tappan Morris, "Scalable TCP Congestion Control". Doctor of Philosophy Thesis January 1999.

[Moss99b] Robert Tappan Morris, "Scalable TCP Congestion Control". Proc. IEEE INFOCOM 2000, pp. 1176 - 1183

[MoWa00] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control", IEEE/ACM Transactions on Networking, Vol. 8(5), pp 556-567, October, 2000.

[OtLa99] Teunis J. Ott, T. V. Lakshman, Larry H. Wong. “SRED: Stabilized RED”. INFOCOM 1999, pp. 1346-1355

[OPNET] OPNET Inc.,“OPNET Modeler Manuals”. OPNET Modeler Version 7.0, 2001

[PeBa01] Pentikousis, H. Badr, and B. Kharmah, *TCP with ECN: Performance Gains for Large Transfers*, SBCS-TR-2001/01, Department of Computer Science, SUNY Stony Brook, March 2001.

[PeDa00] L. L. Peterson, B. S. Davie. “Computer Networks, A Systems Approach”. 2nd Ed., Morgan Kaufmann, San Francisco, 2000[Post81a] J. Postel. “Transmission Control Protocol”. RFC 793. 1981.

[Post81b] J. Postel. “Internet Control Message Protocol”. RFC 792. 1981.

[RaFl01] L. Ramakrishnan, Sally Floyd, D. Black. "RFC 3168: The addition of Explicit Congestion Notification (ECN) to IP". IETF Network Working Group Recommendation for Standards, September 2001. ftp://ftp.isi.edu/in-notes/rfc3168.txt

[SaAh00] J. Hadi Salim, U. Ahmed. “Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks” RFC 2884. July 2000.

[Stev94] W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley, 1994

[Stev97] W. Richard Stevens. "RFC2001 TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms". January 1997. ftp://ftp.isi.edu/in-notes/rfc2001.txt

[Vard01] Dimitrios Vardalis. “On the Efficiency and Fairness of TCP over Wired/Wireless Networks”. Master of Science Thesis Report, Computer Science, State University of New York. 2001

[Wing99] Ng Ping Wing. “An Empirical Study of TCP’s Fairness”. MSc Thesis, Hong Kong Polytechnic University. April 1999.

[YaYa02] Kazunori Yamamoto, Miki Yamamoto, Hiromasa Ikeda. “Congestion Control for Reliable Multicast Achieving TCP Fairness”. IEICE Trans. Communications., Vol. E85-B, No. 1. pp. 183-190. January 2002

[Zhen01] Zici Zheng, “Adaptive Explicit Congestion Notification for Hetrogeneous Flows”, Master of Science Thesis Report, Computer Science, Worchester Polytechnic Institute, May 2001.

[ZhYa01] Chengyu Zhu, Oliver W.W. Yang, James Aweya, Michel Oullette, Delfin M. Montuno. "A Comparison of Active Queue Management Algorithm Using OPNET Modeler". Presented in OPNET 2001, August 2001, Washington DC, USA

Appendix A: TCP/IP Networks Congestion Control

Congestion in TCP/IP networks focused on a phenomenon called "Congestion Collapse". As the load on the network increases, the throughput, queuing delay, and loss rate also increases. Increases in loss and delay cause increases in retransmission rates. These retransmissions in turn increase loss and delay, as well as consuming bandwidth to no useful purpose. In the last decade, several variants of TCP (Tahoe [Jaco88], Vegas [BrMO94], Reno [Jaco90], NewReno [FaFl96], SACK [FaFl96, FlMa00] etc.) have been developed and evaluated to provide the host-centric mechanisms to combat high packet loss rates during heavy congestion periods. Meanwhile, new congestion avoidance techniques for Internet Routers have also been proposed and evaluated. This section describes the end-to-end congestion control issues and different congestion control mechanisms.

A.1 End-to-End Congestion Control Issues

TCP is the dominant transport protocol used in the Internet today. While the Internet traffic continues to grow, it becomes more challenging to provide fair goodput to millions of web users. The dropped packets consume bandwidth on the way to the place they are dropped, and they also cause increased incidence of TCP timeouts. In extreme cases, the situation could lead to Congestion Collapse [Jaco88]. The end-to-end congestion control mechanisms of TCP have been critical factor in the robustness of the Internet.

Congestion Collapse occurs when an increase in the network load results in a decrease in the useful work done by the network. During the mid 1980s, the congestion collapse [Jaco88], also called "Internet meltdown" was first observed. This was largely due to TCP connections unnecessarily retransmitting packets that are in transit or had already been received by the receiver. In order to fix the congestion collapse problem, Jacobson [Jaco88, Jaco90] developed the congestion control mechanisms. These mechanisms operate in the end hosts to cause TCP connections to back-off during congestion. Originally, TCP included window based flow control mechanisms [Stev97]

as a means for the receiver to control the amount of data sent by the sender. The overflow mechanism was used to prevent the overflow of the receiver's data buffer space available for the TCP connection. With the new congestion control mechanism the TCP flows are responsive to the congestion signals (e.g., packet loss) from the network.

Problems with the Congestion Collapse have generally been corrected by the timer improvements and congestion control mechanisms in the modern implementations of the TCP. However, with the expending use of Internet by users and applications, it is no longer practical to rely on all end-nodes to use end-to-end congestion control for best effort traffic. Similarly, it is no longer possible to rely on all developers to incorporate end-to-end congestion control in their Internet applications.

The congestion control algorithms used by TCP results in unfair bandwidth allocation when multiple connections share a congested gateway. TCP/IP networks have a bias against connections passing through multiple congested gateways, a bias against connections with longer roundtrip times, and a bias against bursty traffic [Floy91a]. Several techniques for measuring fairness in networks with congested gateways have been proposed [Jain91, PeBa00]. The issue of fairness among competing flows has become increasingly important with the growth of the Web. Internet users increasingly want high-bandwidth, low delay communications, rather than the leisurely transfer of a long file in the background. The growth of the best effort traffic that doesn't use TCP underscores this concern about fairness between best-efforts traffic in the times of congestion. For the current Internet environment, where other best efforts traffic (e.g. UDP) could compete, at a router queue, with TCP traffic, the absence of fairness could lead to one flow starving out another flow in the time of congestion.

A.2 Traditional Congestion Control Algorithms

Congestion can occur when data arrives on a big pipe (e.g. LAN) and gets sent out to a smaller pile (e.g. WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs. This section describes the traditional Congestion Control algorithms for TCP/IP networks. These algorithms are included in most of the modern implementations of TCP.

Sliding Window Protocol

Sliding Window [Stev97] protocol leads to faster data transfer, since the sender doesn't have to stop and wait for an acknowledgement each time a packet is sent. Figure A-1 summarizes the sliding window protocol.

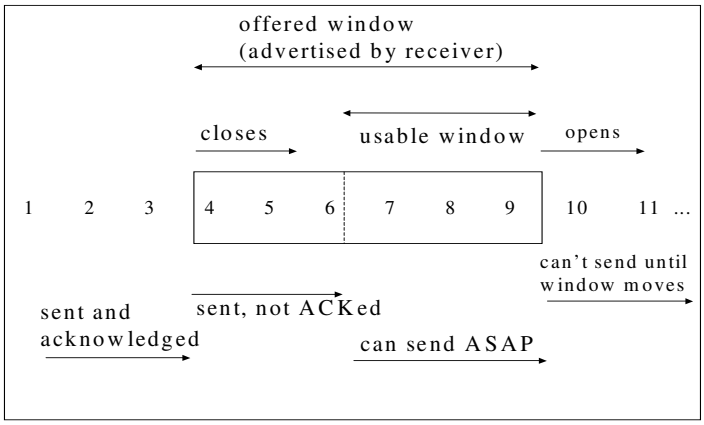


Figure A-1 Sliding Window Protocol

The size of the window offered by the receiver can usually be controlled by the receiving process. The size of the window can affect the TCP performance. While using a small window size may cause under utilization, using window larger than necessary also causes problems. Network routers must buffer the excess packets, causing either excessive delay or buffer overflow and packet loss. In general the correct window size is the product of available capacity (or bandwidth) and round-trip propagation delay. However, TCP cannot directly determine either factor of the delay-bandwidth product, so it uses an adaptive "congestion window" algorithm, which effectively searches for the maximum reasonable window by increasing it until the network starts dropping packets. The algorithm comes in two parts, called slow start and congestion avoidance.

Retransmit Timers

The TCP sender sets a retransmit timer to determine when a packet has been dropped in the network. When the retransmit timer expires, the sender assumes that a packet has been lost, it sets *ssthresh* to half of current window (W), and goes into slow start, retransmitting the lost packets. If the retransmit time expires because no

acknowledgement has been received for the retransmitted packet, the retransmit timer is also backed-off, that is, doubling the value of the next retransmit timeout interval.

Slow Start

TCP Slow Start algorithm [Jaco88, Stev97] maintains the rate at which new packets are injected into the network to the rate at which the acknowledgments are returned by the other end. Slow Start adds another window to the sender's TCP: the congestion window (*cwnd*).

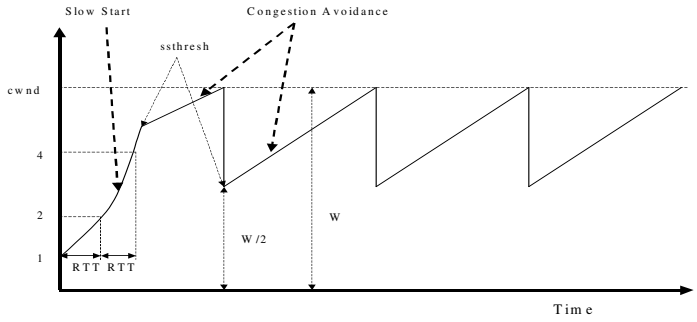


Figure A-2 Slow Start and Congestion Avoidance

Slow start, as shown in Figure A-2, works as follows:

- When a new connection is established with a host on another network, the congestion window is initialized to one segment.
- Each time an ACK is received, the congestion window is increased by one segment. This doubles *cwnd* every round-trip time.

cwnd is maintained in bytes, but slow start always increments it by the segment size. The sender can transmit up to a minimum of the congestion window and the advertised window. The congestion window is the flow control imposed by the sender while the advertised window is the flow control imposed by the receiver.

Congestion Avoidance

TCP in Congestion Avoidance [Jaco88, Stev97]] mode is searching for a reasonable window size. The goal is to increase the window slowly in case available network bandwidth has increased, perhaps because of the reduced competition from other

connections but to detect reductions in bandwidth and reduce the window accordingly. TCP maintains a guess at the current reasonable window size, called the slow start threshold (*ssthresh*).

Congestion avoidance, as shown in Figure A-2, works as follows:

- *cwnd* starts at *ssthresh* due to slow start
- After each entire window of positive ACKs arrive, increase *cwnd* by one packet.
- After a timeout (that is a lost packet), set *ssthresh* to *cwnd*/2 and enter slow start

The algorithm has the effect of increasing the window by one packet per round trip time when there is no loss. The rate at which TCP sends data is equal to one window per round trip time, or *cwnd*/*rtt*. As long as *cwnd* is less than the delay-bandwidth product, increase in *cwnd* causes increase in send rate, because *rtt* is fixed at round trip propagation delay. However, once *cwnd* exceeds the delay-bandwidth product, routers must buffer the excess packets. The buffering increases the round-trip time by one packet transmission time per buffered packet. If we measure *rtt* in units of packet transmission times, we can see that increases in *cwnd* leave *cwnd*/*rtt* unchanged once *cwnd* is equal to the delay-bandwidth product. This discussion provides the following two design considerations [Morr99a, Morr99b] for the routers:

- Router must provide about one delay bandwidth of buffering if it wishes to make sure that a single TCP can sustain a send rate equal to the link bandwidth.
- TCP will tend to keep router buffers full no matter how large they are. This means that building routers with huge buffer will result in excessive queuing delay.

Fast Retransmit and Fast Recovery

TCP generates an immediate ACK (a duplicate ACK) when an out of order segment is received. Since it is not known whether a duplicate ACK is caused by a lost segment or just a reordering of segments. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP's Fast Retransmit [Jaco88, Stev97] algorithm performs a retransmission of what appears to be the missing segment (after three duplicates ACKs), without waiting for a retransmission timer to expire.

With the TCP's Fast Recovery {Jaco90, Stev97] algorithm the source does not slow start (set *ssthresh* to *cwnd*/2 and *cwnd* to 1) after inferring that a packet has been

dropped. Instead, the TCP source effectively waits half a round trip time and halves the congestion window. The source retransmits the dropped packet and uses incoming duplicate ACKs to clock additional outgoing packets.

These algorithms are usually implemented together as follows:

- When the third duplicate ACK is received, set *ssthresh* to one-half of the minimum of the *cwnd* and the receiver's advertised window.
- Retransmit the duplicate segment.
- Set *cwnd* to *ssthresh* plus 3 times the segment size.
- Each time another duplicate ACK arrives, increment *cwnd* by the segment size and transmit a packet (if allowed by the new value of *cwnd*)
- When the next ACK arrives that acknowledges new data (this ACK should acknowledge all the segments sent between the lost packet and the receipt of the third duplicate ACK), set *cwnd* to *ssthresh* (the value set in step 1).

A.3 TCP Variants

Earlier TCP implementations followed a go-back-n model using cumulative positive acknowledgement and requiring a retransmit timer expiration to re-send data lost during transport. These TCPs did little to minimize network congestion [FaFl96]. Various enhancements to TCP have been proposed and implemented to control network congestion while maintaining good user throughput. This section describes the popular TCP variants: Tahoe, Reno, New-Reno, SACK and Vegas.

Tahoe TCP

The Tahoe TCP implementation (distributed with 4.3 BSD Unix) added a number of new algorithms and refinements to earlier implementations. The new algorithms include Slow-Start, Congestion Avoidance, and Fast Retransmit [Section A.2]. This implementation modifies the round trip time estimator to set retransmission timeout values.

Reno TCP

The Reno TCP implementation retained the enhancements incorporated into Tahoe but modified the Fast Retransmit algorithm to include Fast Recovery [Section A.2]. Reno's Fast Recovery algorithm is optimized for the case when a single packet is dropped from a window of data. The Reno sender retransmits at most one dropped packet per round-trip time. Reno significantly improves upon the Tahoe TCP when a single packet is dropped from a window of data, but can suffer from performance problems when multiple packets are dropped from a window of data [FaFl96].

New-Reno TCP

The New-Reno TCP [FaFl96] includes a small change to the Reno algorithm at the sender that eliminates Reno's wait for a retransmit timer when multiple packets are lost from a window. The changes concerns the sender's behavior during Fast Recovery when a partial ACK is received that acknowledges some but not all of the packets that are outstanding at the start of that Fast Recovery period. In Reno, partial ACKs take TCP out of Fast Recovery by increasing the window size back to the size of the congestion window. In New-Reno, partial ACKs do not take TCP out of Fast Recovery. When multiple packets are lost from single window of data, New-Reno TCP can recover without a retransmission timeout, retransmitting one lost packet per round-trip time until all of the lost packets from that window have been retransmitted. New-Reno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated have been acknowledged.

SACK TCP

The SACK TCP [FaFl96, FlMa00] introduces a selective acknowledgement option to allow the receiver to report the non-sequential data that they have received. The main difference between Reno and SACK is in the behavior when multiple packets are dropped from one window of data. SACK augments the cumulative acknowledgement with additional information that allows the receiver to inform the sender which packets have missed. With this information the TCP senders can make more intelligent decision in determining which packets have been lost and in identifying which packets should be re transmitted.

Vegas TCP

The Vegas TCP [BrOM94] uses source-based anticipation of congestion by monitoring the expected and actual (i.e. measured) throughput to improve TCP congestion control. It is reported that it gives 40-70% better throughput than Reno.

A.4 Active Queue Management

Drop-Tail scheme is used by most of the available routers. The arriving packets are stored in the FIFO queue for the particular output port. These packets are removed from the queue after transmission. Each queue has a limit on the number of packets that it can store. If a packet arrives and the queue is full then the packet is dropped.

Drop-Tail Queuing has following problems with TCP traffic:

- It tends to drop the packets from each connection using the queue. This causes "global synchronization" of connections' window size decreases, leading to under-utilization [Morr99a].
- The packet drop with drop-tail mechanism is not independent of connection. TCP's ACK feedback mechanism causes repeating patterns or "phase effects" [FlJa92] in packet arrivals, which can cause different connections to experience different loss rate.
- TCP sources reduce their transmission only after losing packets. Therefore, even with techniques such as congestion avoidance, slow start, fast retransmit and fast recovery mechanism, the performance of TCP congestion control algorithm over drop-tail networks is inadequate in a heavily loaded network.

A variant of drop-tail called Random-Drop [Mank97] discards a randomly selected packet from the router queue instead of the packet that arrives when the queue is full. With this technique the arriving packet is queued. This approach helps eliminate phase effects but does not avoid global synchronization [FlJa93].

AQM proposes a mechanism for preventing packet loss due to buffer overflow. The basic idea behind AQM is to convey congestion notification early enough to the senders, so that senders are able to reduce the transmission rates before the queue overflows and any sustained packet loss occurs. It also eliminates global synchronization and phase effects of drop tail's mechanism. Random Early Detection (RED [FlJa93], an

active queue management algorithm, is recommended by IETF for deployment in IP routers/network. Various variants of RED like Dynamic RED (DRED) [AwOu01], Stabilized RED (SRED) [OtLa99], Blue [FeKa99] etc. have been proposed.

A.5 Explicit Congestion Notification

Explicit Congestion Notification (ECN) [Floy94, RaFl01] protocol provides the mechanism for using Congestion Experienced (CE) code point in the packet header as an indication of congestion, instead of relying solely on packet drops. ECN can be used as a congestion indication policy of AQM mechanisms (e.g. RED, DRED).

Following are the benefits of using ECN in TCP/IP networks:

- Avoid unnecessary packet drops and therefore avoids unnecessary delay for packets.
- With ECN, the congestion notification is promptly received by the TCP source, and the connection does not remain idle, waiting for the TCP retransmit timer to expire, after a packet has been dropped.

Appendix B: Simulation results

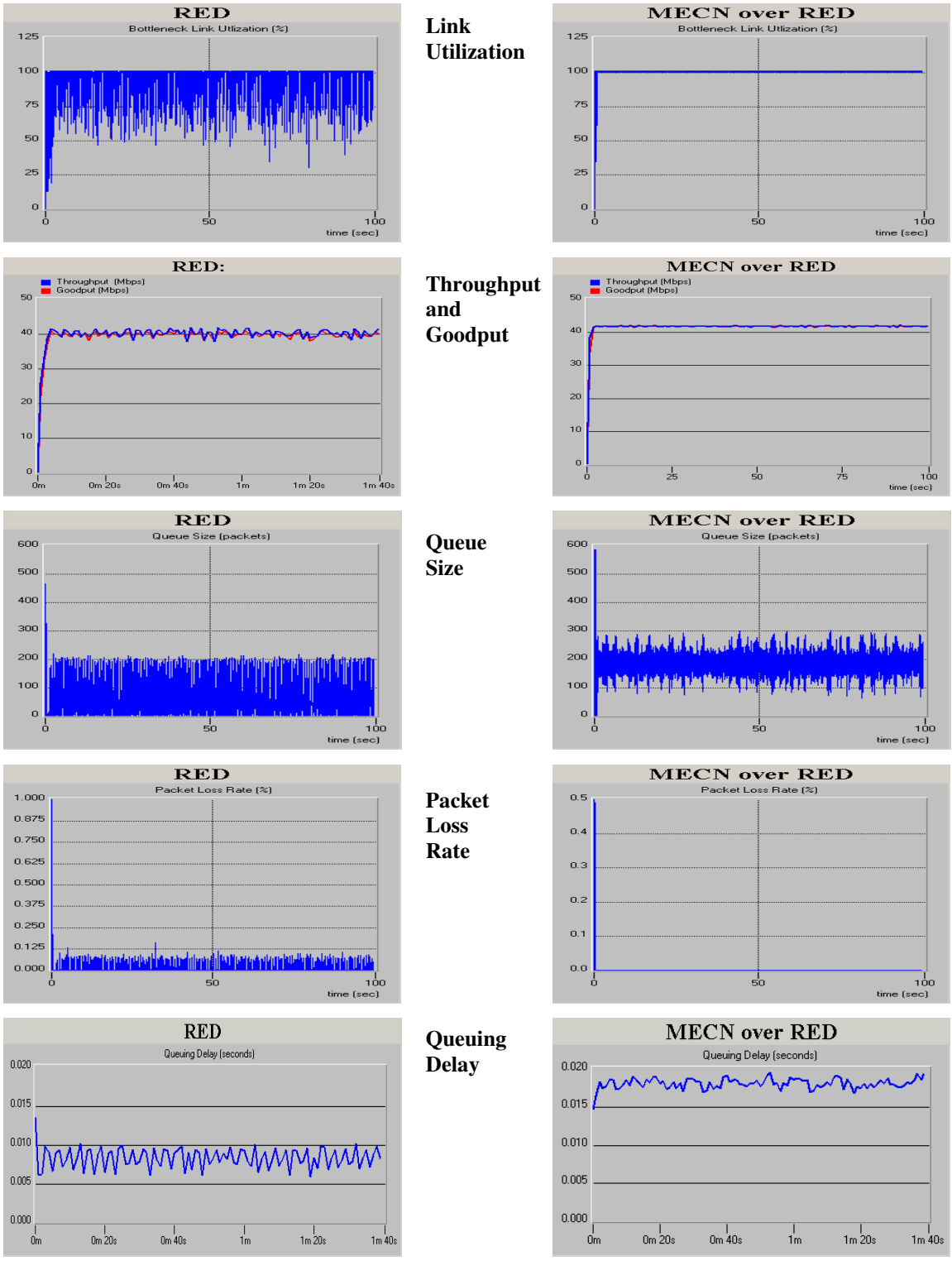
B.1 MECN over RED

Figure B-1, Figure B-2, Figure B-3 show the performance metrics (link utilization, throughput, goodput, queue size, packet loss rate and delay) for “RED” and “MECN over RED” for 100, 500 and 1000 sources respectively. Recall that MECN differs from ECN in that MECN marks packets when the average queue size exceeds max_{th} and drops packets only when the router queue overflows.

The graphs show that the advantages of “MECN over RED” for goodput and link utilization are similar to “ECN over RED”. The difference between throughput and goodput represents the number of retransmit packets. For RED, the number of retransmit packets increases with the increase in the number of sources. This rate of increase is much lower with “MECN over RED”. Goodput i.e. the effective data rate is also higher for “MECN over RED” compared to the RED. However there is no significant improvement in link utilization or goodput by using “MECN over RED” compared to “ECN over RED”.

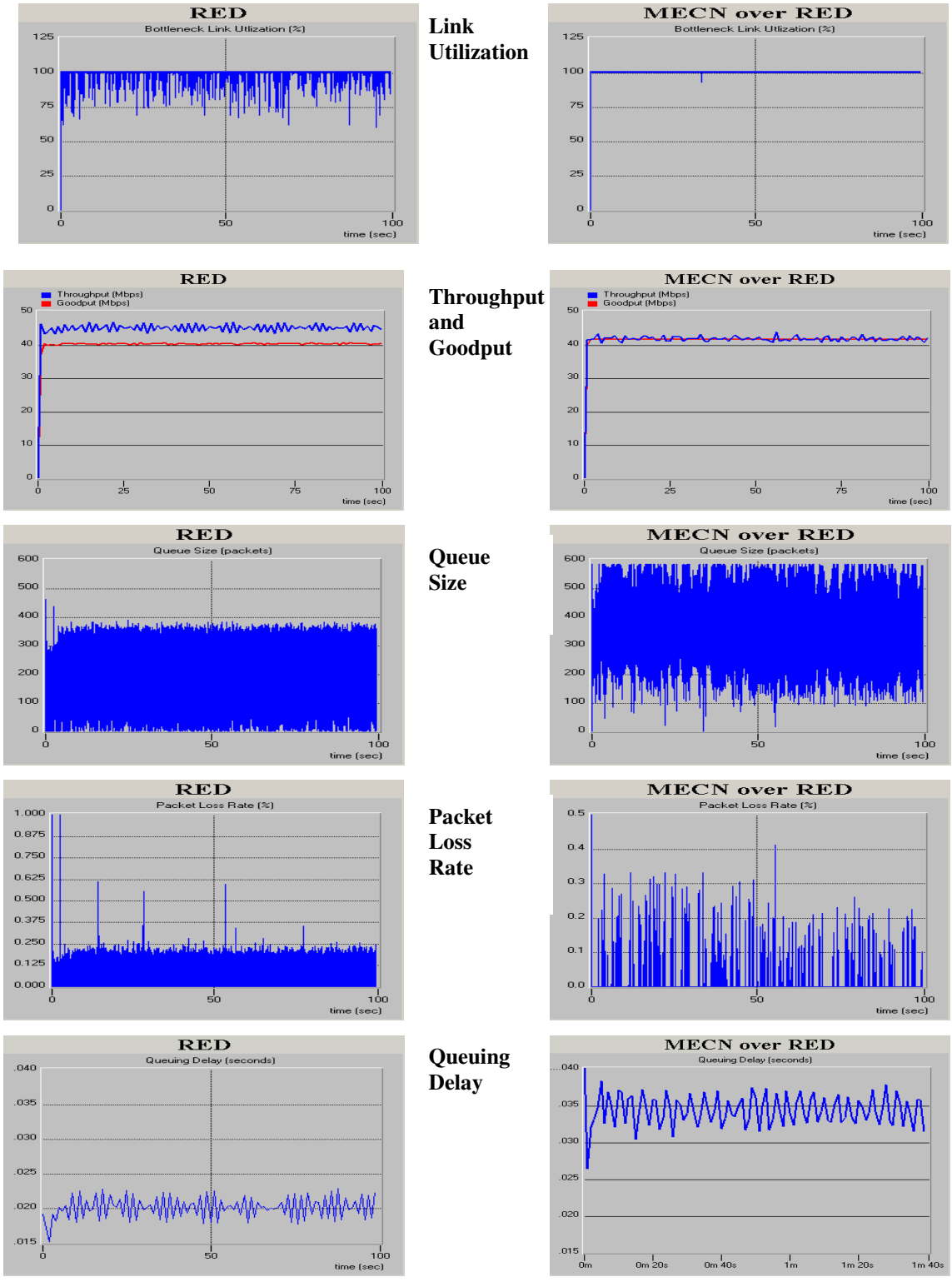
The graphs also show that the increase in queue size for “MECN over RED”. This is because of the extra packets that are marked and queued instead of dropped due to congestion, for "MECN over RED” protocol. By comparing these graphs with the graphs of “ECN over RED” in Figure 3-2, Figure 3-3 and Figure 3-4, increase in queue size for “MECN over RED” is even higher than the increase for “ECN over RED”. This result in frequent buffer overflows and higher queuing delay. Graphs for 500 and 1000 connections clearly show this effect.

Similar to “ECN over RED”, there is a clear need to tune the parameters depending upon the topology and traffic loads for "MECN over RED"



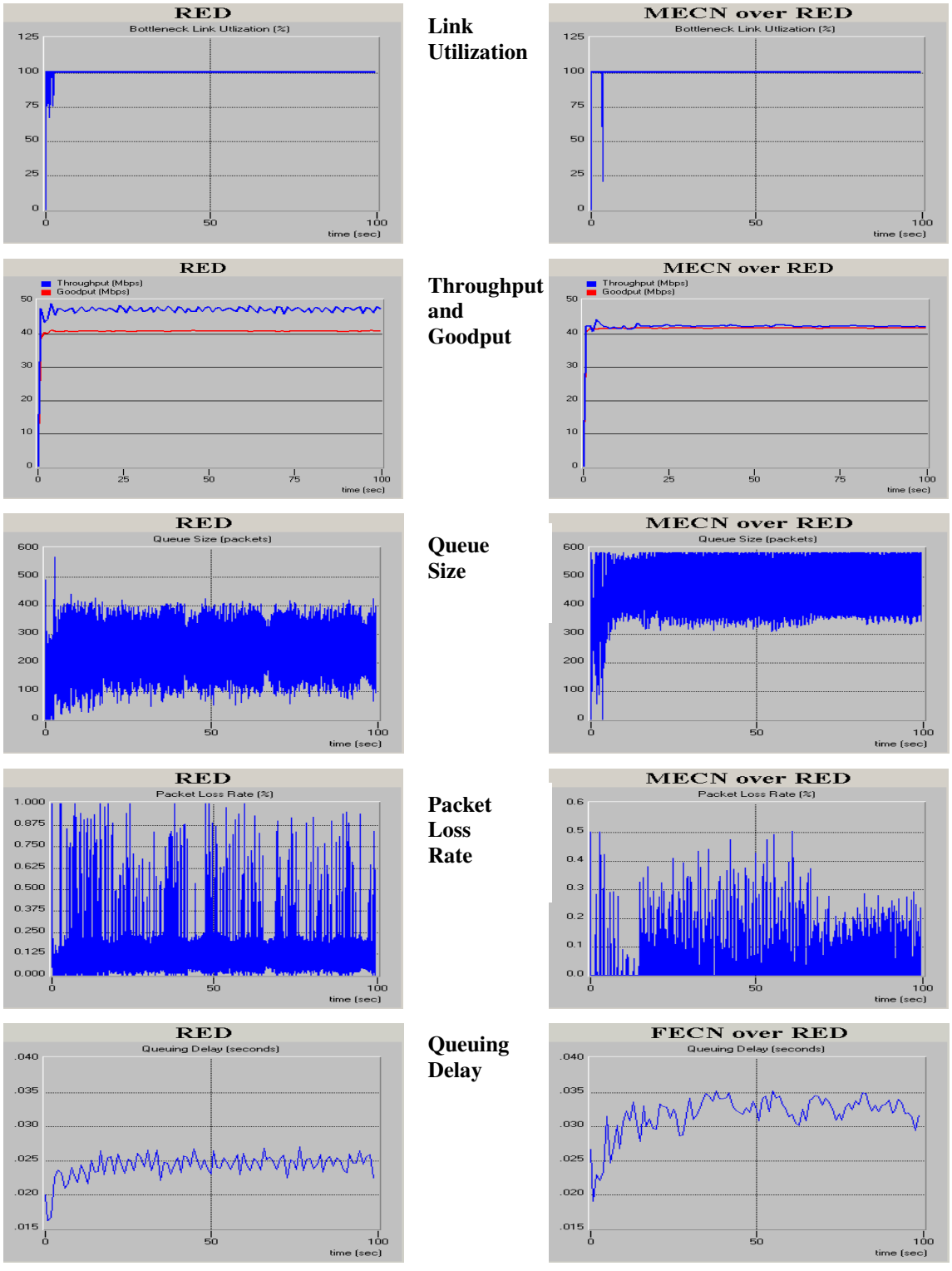
(a) RED **(b) MECN over RED**

Figure B-1 RED and "MECN over RED" Performance Characteristics (100 Sources)



(a) RED **(b) MECN over RED**

Figure B-2 RED and "MECN over RED" Performance Characteristics (500 Sources)



(a) RED **(b) MECN over RED**

Figure B-3 RED and "MECN over RED" Performance Characteristics (1000 Sources)