NOTE TO USERS

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Xintong Liu

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE / DEGREE

Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Different Techniques for Congestion Avoidance Algorithms in Bottleneck Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Oliver Yang

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Wail Gueaieb

Shervin Shirmohammadi

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Different Techniques for Congestion Avoidance Algorithms in Bottleneck Networks

By

Xintong Liu

A thesis submitted to the
School of Graduate Studies and Research
In partial fulmilment of the requirements for the degree of

Master of Science

Master Program in Systems Science
Faculty of Administration
University of Ottawa

May 9, 2005

Your file Votre référence
ISBN: 0-494-11331-6
Our file Notre référence
ISBN: 0-494-11331-6

Abstract

In this thesis, we systematically examine the ECN (Explicit Congestion Notification) mechanism to be applied to some AQM (Active Queue Management) techniques developed in recent years, namely, RED (Random Early Detection), BLUE, ARED (Adaptive RED) and PI-RED (Proportional and Integral RED). We use packet marking as the indication of network congestion, instead of the strategy of dropping packet currently adopted by these AQM algorithms. The performance comparison of ECN-AQM with AQM has been conducted in not only a single bottleneck network but also a multi-bottleneck network environment. The simulation results have shown that ECN-AQM algorithms can greatly decrease unnecessary packet loss, one of the main shortcomings of RED and some other AQM algorithms. They can also reduce average queue size in most cases, but the queue size oscillation problem remains.

We therefore investigate the application of the head dropping policy to AQM techniques, as a simple solution to the problem of queue oscillation, another shortcoming of RED and its variants. With this method, instead of tail dropping, which is currently used by RED and many other AQM schemes, the TCP source can be informed of the congestion occurring in the bottleneck router earlier by getting rid of time to wait through the queuing delay. We have compared DH-RED (Drop Head RED) and DH-BLUE (Drop Head BLUE) with the current RED and BLUE in both of the single bottleneck and the multi-bottleneck networks. We found the performance of queue size stability can be greatly improved by DH-RED and DH-BLUE.

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Oliver Yang, for his research guidance and suggestions throughout the research. I am also thankful to the members of the CCNR lab. It is pleasure to study and work with them.

Finally, I would like to express my deep gratitude to my family, my mom and dad, especially my husband Jun and my son Tongtong, for their endless support and love.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

		Section of 1st appearance
ACK	Acknowledgement	1.2.1
AQM	Active Queue Management	1.1.0
ARED	Adaptive RED	1.1.0
ATM	Asynchronous Transfer Mode	1.2.2.2
CE	Congestion Experienced	3.1.0
CWR	Congestion Window Reduced	A5
DfF	Drop from Front	1.2.2.1
DH-BLUE	Drop Head BLUE	1.5.0
DH-RED	Drop Head Random Early Detection	1.5.0
DRED	Dynamic RED	1.2.2.1
DT	Drop Tail	1.2.2.1
ECE	Explicit Congestion Notification-echo	A5
ECN	Explicit Congestion Notification	1.1.0
ECT	ECN-capable Transport	A5
ERD	Early Random Drop	1.2.2.1
FACK	Forward Acknowledgement	1.2.1
FIFO	First in First out	1.2.2.1
FRED	Fair RED	1.2.2.1
FSM	Finite State Machine	2.4.0
FTP	File Transfer Protocol	2.3.2
HTTP	Hypertext Transfer Protocol	2.3.1
IETF	Internet Engineering Task Force	1.1.0
ISP	Internet Service Provider	2.2.0
LAN	Local Area Network	2.2.0
OPNET	Optimized Network Engineering Tool	1.3.0
PI	Proportional and Integral	1.2.2.1
PIR	Proportional and Integral Rate	1.2.2.2
PI-RED	Proportional & Integral Controller in RED Router	1.1.0
QoS	Quality of Service	1.2.0
RED	Random Early Detection	1.1.0
RTT	Round-Trip Time	1.2.1
SACK	Selected Acknowledgement	1.2.1
TCP/IP	Transmission Control Protocol / Internet Protocol	1.1.0
WAN	Wide Area Network	2.2.0

List of Notations and Symbols

		Section of 1st appearance
B	Buffer size	A2
Freeze_time	Minimum time interval for updating the drop probability in BLUE	3.2.0
N	Traffic Load factor	3.2.0
p	Drop probability assigned to the incoming packet in PI-RED	A4
p_b	Temporary packet drop probability in RED	A1
p_h	Parameter of think time probability distribution	2.3.1
p_l	Parameter of think time probability distribution	2.3.1
$p_{\max}$	Maximum allowed value for p_b, maximum packet drop probability	3.2.0
p_0	Reference packet drop probability in PI-RED	A4
q	Current queue size	A1
q_0	Reference queue size	3.2.0
T	Queue size control target	3.2.0
R_0	Round trip time limit	3.2.0
Sampling_interval	Sample frequency in PI-RED	3.2.0
avg	Average queue size	A1
count	Number of packets that have been forwarded since last dropped packet	A1
cwnd	Congestion Window	1.2.1
d1	Increase drop probability in BLUE	3.2.0
d2	Decrease drop probability in BLUE	3.2.0
p(0)	Initial packet drop probability in BLUE	3.2.0
ssthresh	Slow Start Threshold	1.2.1
$th_{\min}$	Minimum queue threshold	3.2.0
$th_{\max}$	Maximum queue threshold	3.2.0
w_q	Queue weight, queue averaging filtering gain	3.2.0
α	Parameter of file length probability distribution	2.3.1
α_{ARED}	Increment parameter for maximum packet drop probability in ARED	3.2.0
β_{ARED}	Decrement parameter for maximum packet drop probability in ARED	3.2.0
δp	PI controller output	A4
δq	Queue size regulation error, the input of the PI controller	A4

Δt	Sampling interval in ARED	3.2.0
η_h	Parameter of think time probability distribution	2.3.1
η_l	Parameter of think time probability distribution	2.3.1
π	Parameter of file length probability distribution	2.3.1
π_h	Parameter of think time probability distribution	2.3.1
π_l	Parameter of think time probability distribution	2.3.1

Chapter 1 Introduction

1.1 Overview

The development of computer technology has made striking progress in a short time compared with other industries. With the merging of computer and communications, especially the evolution of the Internet, computer networking has experienced a booming growth, which has required high efficient traffic engineering mechanisms to manage the traffic exposed to networks in order to provide ever-increasing users with fair and reliable services. TCP/IP (Transmission Control Protocol / Internet Protocol) [Stev94], as the dominant network traffic control protocol suite, allows computers of all sizes and from many different computer vendors to communicate with each other while running totally different operate systems. In particular, its congestion control and avoidance mechanism has been playing an important role in preventing networks from degradation and congestion collapse since the mid 1980s when the congestion collapse [Jaco88], otherwise known as "Internet meltdown" was first observed. For the current Internet environment, the TCP-Reno algorithm [Jaco90], the most-adopted traffic control algorithm working in TCP sources and destinations, cooperates with the RED[1] (Random Early Detection) algorithm [FlJa93], the most popular AQM (Active Queue Management) algorithm working in routers, to govern network traffic and avoid the occurrence of congestion.

Although RED is the only deployable AQM algorithm advocated by IETF (the Internet Engineering Task Force) in the current networks, its two problems are obvious: queue size oscillation and unnecessary packet loss. Both will lead to poor network performance such as large delay and low link utilization. To overcome the shortcomings of the RED algorithm, some new AQM and TCP techniques were proposed in recent years, such as ECN (Explicit Congestion Notification) [Floy94], BLUE [FeKa02], ARED (Adaptive RED) [FeKa99, FlGu01], and PI-RED (Proportional and Integral Controller in RED capable Routers) [HoMi01b]. Each of these algorithms, on the one hand, has its own strong points, which improve network performance to some extent; on the other hand, exhibits various

[1] Both of the terms "Random Early Detection" and "Random Early Drop" have been seen in literatures. However, following the author who proposed the RED algorithm, we use the former in this thesis.

performance problems under different network environments. Considering that the networking service demands increase at an exponential rate and the bandwidth available to users is limited, the network traffic and congestion control mechanism will still be an essential technology for the quality of Internet services, which deserves further study.

1.2 Related Work

Since a large number of widely dispersed Internet sites experienced the "congestion collapse" predicted by Nagle [Nagl84] (i.e., a simultaneous slowdown or cessation of networking services for prolonged periods) during several periods of 1986 and 1987, the gateway router congestion encountered by Internet users has become an obstacle for Internet applications with the increasing growth of Internet traffic. This has made the congestion control and avoidance mechanism a critical factor in the robustness of the Internet.

In fact, the congestion control and avoidance mechanism has drawn a lot of attention from researchers over a decade. Generally, the major traffic and congestion control algorithms regarding TCP/IP networks, which have been proposed so far, can be grouped into two categories according to where the congestion control information is obtained from and where the algorithms are implemented. The first group is end-host-based (TCP) congestion control, in which a TCP source regulates its window size based on the information obtained from the destination, and congestion control algorithms are implemented in the source. The second group is router-feedback-based congestion control, in which a TCP source controls traffic based on the information fed back from the router, and the algorithms are implemented in the router. Usually, the TCP control and the router control co-exist in the network traffic control mechanism in order to meet the requirements of QoS (Quality of Service).

1.2.1 End-host-based (TCP) Congestion Control

The end-host-based (TCP) congestion control [Jaco88, Jaco90, Stev97] uses a window-based flow control mechanism to effectively regulate the transmission rate of individual connection according to the level of congestion in the network. The basic idea behind the end-host-based (TCP) congestion control mechanism can be briefly summarized as follows. A TCP source regulates network load by adjusting its own congestion window (*cwnd*) size to limit the

amount of data it can transmit into the network before receiving an ACK (acknowledgement) in response to the level of congestion in the network. Upon receipt of successful ACKs, the source increases its *cwnd* size in order to increase its transmission rate, and then sends out packets based on the new *cwnd* size after considering the number of the packets still in flight. When the total transmission rate eventually exceeds the network's capacity, a queue builds up in the router. Once the queue overflows, any incoming packet is dropped, and duplicate ACKs will be sent from the destination to the source. If the source receives a given number of duplicate ACKs for a given packet, it then assumes that the packet is lost, and reduces its transmission rate by decreasing its *cwnd* size. If the source does not receive any ACKs for a period of time, it goes into a timeout period by triggering a retransmission timer. Then the *cwnd* size is decreased to one packet and the lost packet is retransmitted. A thorough description of the TCP mechanism can be found in [Stev94, Stev97].

Basically, the TCP congestion control policy contains four parts: *Slow Start*, *Congestion Avoidance*, *Fast Retransmit, and Fast Recovery* [Jaco88, Jaco90, Stev94, Stev97, AlPa99]. At the beginning of TCP implementations, a Go-Back-N model [Sast75], which uses cumulative positive ACKs and retransmission timer expiration to retransmit lost packets, was adopted. Since Van Jacobson and Mike Karels developed a collection of several algorithms including *Slow Start*, *Congestion Avoidance* and *Fast Retransmit* [Jaco88] in 1987, the TCP-Tahoe implementation [Jaco88] has combined these three algorithms and the Go-Back-N model to manage a source's *cwnd* size in response to congestion in the network.

TCP-Reno [Stev94, Stev97, AlPa99], the most-adopted TCP algorithm in the current networks, combined those three algorithms incorporated in TCP-Tahoe, and added a new feature called *Fast Recovery* [Jaco90] in the *Fast Retransmit* operation. With the *Fast Recovery* algorithm, a TCP source halves its *cwnd* size instead of setting it to one after *Fast Retransmit,* and uses incoming duplicate ACKs (usually 3 ACKs) to clock subsequent outgoing packets instead of *Slow Start* in TCP-Tahoe. TCP-Reno's *Fast Recovery* algorithm can improve the efficiency of link utilization when only one packet is discarded in a certain window. However, it suffers from the performance problems in the case that multiple packets are dropped from one window since the TCP source has to wait for retransmit timer expiration before it can resend packets after the first fast retransmission.

Aimed at solving the problems experienced by the TCP-Reno algorithm when multi-packet losses happen in one window, TCP-New-Reno [Hoe96, FlHe99], a variation of TCP-Reno was proposed, with a small change to the TCP-Reno's *Fast Recovery* algorithm. It remains in the *Fast Recovery* state, and retransmits lost packets one by one in each round trip time (RTT) without waiting for the occurrence of retransmission timeout because it assumes that the packets immediately following the acknowledged one have been lost. With TCP-New-Reno, the source's throughput can be increased significantly in the case of multi-packet losses in one window.

In addition to the congestion control schemes mentioned above, there are other techniques that have been proposed in recent years, such as TCP FACK (TCP Forward Acknowledgement) [MaMa96a], TCP SACK (TCP Selected Acknowledgement) [MaMa96b], and TCP-Vegas [BrOM94]. Researchers have done a lot of work on the TCP model and flow analysis, as well as the comparisons of different TCP algorithms [FaFl96, BaPa97, MaSe97, Morr97]. Most of these algorithms, except TCP-Vegas, have a feature that a TCP source reduces its transmission rate after detecting packet losses. Although these TCP algorithms achieve the performance improvement on the network traffic and congestion control, none of them can maintain a relatively stable transmission rate in the source and stabilize the queue size in the router efficiently.

1.2.2 Router-feedback-based Congestion Control

There are two subcategories for router-feedback-based congestion control. One is implicit router-feedback-based congestion control. The other is explicit router-feedback-based congestion control. The algorithms in both of the groups feed the information about the router back to the TCP source.

1.2.2.1 Implicit Router-feedback-based Congestion Control (AQM)

Most implicit router-feedback-based congestion control mechanisms use queue management technologies to control the queue size in the router to an acceptable level by applying an appropriate packet dropping policy at an appropriate time, based on the implicit information (dropped packets) received from the router. A queue management mechanism has been considered a significant role in keeping networks efficient for a long time. Good queue

management techniques can stabilize network traffic and utilize link capacity more efficiently. Otherwise, the router buffer can easily be overloaded or be in an idle state. In the former case, a large number of packets have to be discarded and then require retransmission, which wastes bandwidth, causes the reduction of source transmission rate, and even makes the system idle, resulting in the underutilization of system resources.

Traditionally, queue service is based on the First In First Out (FIFO) policy. Packets are stored in the queue when they arrive, and are removed from the queue after being transmitted. If a packet arrives and the queue is full, then the packet is discarded. Each queue has a limit on the number of packets that it can hold.

There are several queue management algorithms that have been implemented in the networks, such as DT (Drop Tail), DfF (Drop from Front) [YiHl93, LaNe96], ERD (Early Random Drop) [Hash89] and some other popular AQM algorithms. DT is the simplest and most commonly used congestion control algorithm in the Internet. It drops any arriving packet from the tail of the queue if it is full. In contrast to DT, DfF makes room for incoming packets by discarding the packet from the head of the queue. Instead of waiting for router queue overflow and having to drop all incoming packets in DT and DfF, ERD discards arriving packets with a fixed packet drop probability when the queue occupancy reaches a certain dropping threshold. Although ERD performs better than DT and DfF in terms of packet loss rate, it is not suitable for bursty sources due to its fixed packet drop probability.

The performance of the above queue management algorithms is not good enough for the heavily loaded network to achieve high QoS, even with the techniques like TCP congestion window, *Congestion Avoidance*, *Slow Start*, *Fast Retransmission* and *Fast Recovery*. As one of the solutions to the problems of these algorithms, AQM techniques were proposed to inform the TCP source of possible congestion by proactively discarding incoming packets usually from the tail of the queue according to the output of a random probability function. Hence the TCP source can get this incipient congestion information to reduce its transmission rate before congestion really happens.

As the most well-known and the only deployable AQM algorithm recommended by IETF, RED [FlJa93] has attracted a lot of researchers to work on its theoretical analysis and performance evaluation [FiBo00, TiMa01, HoMi01a, KuLa01, LoPa02] since first being proposed by S. Floyd and V. Jacobson in 1993. RED uses average queue size to measure

traffic load, and prevents congestions by randomly discarding incoming packets with a certain probability (a function of the average queue size) when incipient congestion is detected in the network. The detailed description of the RED algorithm is summarized in Appendix A.1. RED has the advantages of avoiding congestion, global synchronization of sources and biases against burst connections and maintaining a relatively high efficiency of link utilization. While RED outperforms its predecessors significantly, its two problems are obvious. First, early congestion notification relies on the unnecessary loss of packets, which not only wastes network resources but also may lead to network collapse, or the whole network shutdown, due to serious packet loss. Second, the queue size under the control of RED oscillates dramatically, which results in the network to be unstable. This oscillation is an inevitable result of the protocol itself [FiBo00, HoMi01a], and it is extremely hard to relieve the oscillation by tuning RED parameters [BoMa00, ChJe00]. It is also revealed that RED becomes unstable when the round trip delay of TCP link becomes large [LoPa02], and such delay can be caused by the packet loss imposed by IP networks [Morr00].

Since the proposal of RED, a number of modifications and alternatives to RED have been introduced to overcome the inherent weaknesses of RED, such as BLUE [FeKa02], ARED [FeKa99, FlGu01], and PI-RED [HoMi01b].

Aimed at reducing the unnecessary packet loss experienced by RED and some other algorithms, BLUE (Ref: Appendix A.2) was proposed to directly use packet loss and link idle, as congestion indicator, rather than relying on queue size to measure congestion. In addition to retaining the advantages of RED in all of above aspects, BLUE has been shown to be superior to RED in decreasing packet loss even when using a smaller buffer.

In 2001 S. Floyd et al. recognized that the average queue size in RED is quite sensitive to the level of congestion and to the setting of the RED parameters, and therefore not predictable. They proposed ARED (Ref: Appendix A.3) [FlGu01] (a revised version of the algorithm firstly proposed by Feng et al. [FeKa97]) to solve these problems, with minimal changes to the original RED algorithm. ARED can maintain a pre-defined target average queue size in a wide variety of situations, and reduce the sensitivity to the parameters that effect the performance of RED by adjusting the parameter of the maximum drop probability in RED and automatically configuring RED's other two parameters, which are queue weight and maximum queue threshold.

Unlike RED whose packet drop probability is randomly determined by a probability function, PI-RED (Ref: Appendix A.4) uses a proportional and integral (PI) controller to calculate its packet drop probability in order to control the average queue size at a reference level. If there is a difference between the average queue size and the reference queue size, then the PI controller will verify its output to change the packet drop probability. PI-RED is shown to outperform RED significantly in terms of a high level of utilization and a low level of latency.

All variants of RED mentioned above, including the RED algorithm, share the basic idea of RED by discarding packets to inform the TCP source of possible congestion before it really happens. In addition, they all focus on seeking and adjusting a suitable packet drop probability to achieve the desired performance goals.

Basically, these algorithms can improve the performance of RED to some extent. However, since packet dropping is the only way to inform of congestion, it is inevitable that performance problems such as unnecessary packet losses and large delays are observed.

1.2.2.2 Explicit Router-Feedback-based Congestion Control

Both end-host-based (TCP) congestion control (e.g. TCP-Reno) and implicit router-feedback-based (AQM) congestion control (e.g. RED) usually co-exist in networks. However, the combination of TCP-Reno and RED cannot achieve an optimal performance because the RED-capable router informs the TCP-Reno source of network congestion only by dropping packets. These dropped packets, on the one hand, are implicit information, which cannot efficiently help the source adjust its congestion control actions, and on the other hand, consume link bandwidth, and therefore increasing unnecessary timeouts, and introducing large delays.

Many researchers have noticed this problem and have proposed an alternative strategy: the explicit router-feedback-based congestion control mechanism. It can explicitly notify a TCP source of congestion, either with a single digital bit as a flag indicating the router congestion status, or with the exact bandwidth available in the router, (e.g. the recommended source transmission rate). Therefore, the source can use this information to adjust its transmission rate in order to avoid possible congestions.

ECN

ECN (Explicit Congestion Notification) (Ref: Appendix A.5) is one of the explicit router-feedback-based congestion control schemes, and proposed by Sally Floyd [Floy94]. It informs a TCP source of possible congestion by marking packets using the Congestion Experienced (CE) bit in the IP header of the packet, instead of by dropping packets, as most AQM algorithms do. This scheme solves the problem of unnecessary packet loss experienced by both of the TCP and the AQM algorithms. It also reduces unnecessary timeouts, and thus provides low-delay services. However, ECN only tells the source whether the router is congested or not, and does not give the information of the exact bandwidth available in the router to help the source explicitly adjust its transmission rate in response to the congestion. The performance analysis of ECN can be found in papers like [RaFl99, KaKa00].

AQM techniques usually use ECN as their alternative congestion indication policy. Performance evaluations of ECN with RED and ECN with BLUE have been investigated through detailed simulations [SaAh00, ArZh02, FeKa02]. However, very limited work has been done on ECN with other variants of RED such as ARED and PI-RED.

Rate-based Congestion Control

Another type of explicit router-feedback-based congestion control is rate-based congestion control (e.g. [KaKa00, KaKu00]), in which the router provides the exact bandwidth available in the router or the recommended source transmission rate to help the TCP source explicitly and efficiently regulate traffic and congestion. By using modern control theory, a proportional feedback-based rate controller was designed to control the traffic flow in both ATM (Asynchronous Transfer Mode) networks [ZhYa97] and TCP networks [ZhYa00], with the stability criteria presented and proved with Lyapunov Second Stability Criterion. This proportional rate controller computes the output of the controller (the recommended source rate) based on the router queue size and the reference queue size, which is then converted into the advertised window size through a window-based calculation, and uses an ACK-like feedback mechanism to convey this window size to the source. The source then adjusts its data transmission rate and affects the queue size in the router to prevent congestion. Under the regulation of this controller, the traffic is quite stable, and link utilization is highly improved. However, it has a limitation of the steady-state error problem [CoZh03]. A new

controller called PIR (Proportional and Integral Rate) controller [CoYa04], based on the continuous-time fluid flow control theory, was proposed with a theoretical proof of its stability. A recommended global source rate, which is different from the one adopted in the proportional rate controller, was used to solve both of the performance problems observed in RED (queue size oscillation and unnecessary packet loss), and the steady state error problem in the pure proportional rate controller. However, PIR and other rate-based congestion control approaches have the limitation that they require the current implementations for both TCP end hosts and gateway routers to be modified significantly.

1.3 Motivation

As reviewed before, one of the two shortcomings of the current TCP-Reno plus RED mechanism is that it relies on the unnecessary loss of packets as an indication of congestion. This can be solved by the introduction of the ECN mechanism, whose importance and benefits have been pointed out by some researchers [RaFl99, HoMi01b]. Another shortcoming of RED is queue size oscillation, and this can be well improved by PI-RED. Hence, it becomes natural to ask whether the combination of ECN and PI-RED can solve both shortcomings in RED, and this would require a systematically ultimate explanation and the performance comparison of ECN-PI-RED with RED.

Of the limited work done in the performance of couple AQM algorithms using ECN (called ECN-AQM in this thesis), there appears to be no comparison with the pure AQM algorithms. Therefore it is hard to appreciate the overall effect of ECN on these AQM algorithms. Specifically, we have noticed that both BLUE and ECN can efficiently keep their packet drop rates low, therefore an interesting topic is that whether the combination of ECN and BLUE can double the performance by reducing the packet drop rate by half. The above observations motivate us to analyze and evaluate the influence of the ECN mechanism on AQM by comparing some ECN-AQM algorithms such as ECN-RED, ECN-BLUE, ECN-ARED, and ECN-PI-RED with the original AQM algorithms i.e., RED, BLUE, ARED, and PI-RED, under a wide variety of simulation scenarios.

So far, all simulation-based investigations regarding both ECN-AQM and AQM algorithms were performed only in a single bottleneck network, which may be limited in practicality, and in the literature there does not appear to be any performance evaluation for

these algorithms in a multi-bottleneck network environment. Therefore, we are very much interested in studying these ECN-AQM algorithms in a multi-bottleneck network, which is more able to reflect the actual network environment. We would also like to compare with the performance in a single bottleneck network

As the third motivation, we have a special interest in the general characteristics of a multi-bottleneck network since most network researchers rarely use the multi-bottleneck network model to study network congestion so far.

As an attractive but simple (with minimum modification to the existing algorithm) solution to the queue oscillation problem, we would like to apply the head dropping mechanism to RED (called DH-RED in this thesis, and the application of DH to AQM is called DH-AQM) in order to reduce the delay of the conveyance of congestion information. We will also apply the same rule to BLUE (called DH-BLUE) to further improve its performance.

Since we have introduced both DH and ECN to solve the main problems of AQM, it would be natural to study the tradeoff between DH-AQM and ECN-AQM through the comparisons of DH-RED with ECN-RED, and DH-BLUE with ECN-BLUE.

1.4 Objectives

Generally, we are interested in understanding how different techniques can be used to solve the problems experienced by their correspondingly individual AQM algorithms. Specifically, we want to

1. Study the effect of ECN on some AQM schemes, both improvements and degradations,
2. Study the effect of DH on some AQM algorithms, both improvements and degradations,
3. Investigate how the studied algorithms perform in both a single bottleneck network environment, and a multi-bottleneck network environment, which is rarely used by network traffic researchers,
4. Compare ECN-AQM with DH-AQM,
5. Explore in details the general features of the multi-bottleneck network,
6. Create a network model package, which includes a single bottleneck network model and a multi-bottleneck network model, and is more convenient to use.

1.5 Methodologies and Approaches

In order to achieve the goals listed in the last section, a lot of simulation work needs to be done. All of the simulations are based on the powerful simulation tool, OPNET [OPNE00]. OPNET is an engineering system capable of simulating large-scale communication networks with detailed protocol modeling and performance analysis. This commercial tool provides a convenient and complete user-oriented procedure for installation and simulation development, which makes it the best choice for our simulation work.

We have developed the whole set of our simulation model in OPNET 9.1, called the upgraded OPNET TCP model. This model package can be divided into three different layers: the network models, the node models and the process models. Our investigated congestion control algorithms can be implemented in the router process models while the TCP protocol is traditionally implemented in both of the source process models and the destination process models. In addition to these hierarchical models, we have created some other models, including the packet format model, the probe model, the simulation sequence and the analysis configuration tool. All these models served for our simulation-based investigation and evaluation.

To study network traffic congestion and evaluate the performances of congestion control algorithms, we create bottleneck routers in our simulation models. In addition to building the common-used single bottleneck network model, we also create the multi-bottleneck network model to see how congestion control algorithms perform in such a non-common-used, but more practical communication environment. Moreover, based on these two models, we create various scenarios, each with the different numbers of long-lived sources and short-lived sources, to achieve a relatively comprehensive understanding of our investigated algorithms.

To allow a fair comparison among our investigated algorithms, we set up the common simulation environment for them, such as the same and reasonable packet size and buffer size, the same propagation delay for each link, and the same simulated time. Also, we set most parameters of the algorithms according to the recommended values in the original papers, in order to ensure a fair comparison study.

All TCP sources in our models are designed to be ECN-enabled or ECN-disabled, which can be easily set by users, in order to provide a flexible way to observe how a certain algorithm performs with or without ECN involvement.

In order to verify the correctness of our upgraded OPNET TCP model, we compare the performance of a certain algorithm in this model with that in its predecessor model, which has already been verified.

Also, we study the general characteristics of our multi-bottleneck network model by observing how a certain algorithm (we have chosen RED as such a algorithm) performs when the situation of router data service rate changes in such a network.

In general, there are five experimental steps taken in this thesis. After building the hierarchical simulation models, we examine some AQM algorithms such as RED, BLUE, ARED and PI-RED in a single bottleneck network configuration, which serves as the base of our performance comparison and will be combined with ECN later. Next we merge ECN into these AQM algorithms, and investigate the effect of ECN on AQM by comparing ECN-RED, ECN-BLUE, ECN-ARED and ECN-PI-RED with RED, BLUE, ARED and PI-RED respectively, through numerous simulations on the performance interests, such as queue size and packet drop rate. Then we evaluate all of these algorithms in a multi-bottleneck network configuration. Finally, we investigate two improved algorithms for RED and BLUE, DH-RED (Drop Head RED) and DH-BLUE (Drop Head BLUE) by comparing DH-RED and DH-BLUE with RED and BLUE, in both of the single bottleneck network and the multi-bottleneck network models.

1.6 Thesis Contributions

The following are contributions to this thesis:

1. Systematical evaluation of the effect of ECN on AQM, through simulation-based performance comparisons of RED with ECN-RED, BLUE with ECN-BLUE, ARED with ECN-ARED and PI-RED with ECN-PI-RED, in terms of instantaneous queue size, average queue size and average packet drop rate, in both a single bottleneck network and a multi-bottleneck network models,

2. Investigation of the application of head dropping to AQM by comparing DH-RED and DH-BLUE with the original RED and BLUE, respectively, in both of the single bottleneck network and multi-bottleneck network models,
3. Analysis of the tradeoff between ECN-AQM and DH-AQM, through the comparisons of ECN-RED with DH-RED, and ECN-BLUE with DH-BLUE,
4. Characterization of the multi-bottleneck network through the observation of RED performance when the situation of router data service rate changes in the network,
5. Development and verification of our upgraded OPNET TCP model set. This model has upgraded the original OPNET TCP model as well as extended the single bottleneck network model to a multi-bottleneck one. All investigated congestion control algorithms, including RED, BLUE, ARED, PI-RED, ECN-RED, ECN-BLUE, ECN-ARED, ECN-PI-RED, DH-RED, and DH-BLUE, are now implemented in both of the models.

1.7 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 presents the network operation, the network configuration, the TCP traffic flow models, the OPNET models and the assumptions made in this thesis. This chapter is the framework for our studies. Chapter 3 describes the ECN-RED, ECN-BLUE, ECN-ARED and ECN-PI-RED algorithm, and evaluates the behaviours of these schemes in a single bottleneck network environment. The simulation-based performance comparison and evaluation for these schemes in a multi-bottleneck network environment are provided in Chapter 4. Chapter 5 investigates two improved algorithms of RED and BLUE, DH-RED and DH-BLUE, and further compares them with ECN-RED and ECN-BLUE. Chapter 6 concludes the thesis and provides some suggestions of topic for future work.

1.8 Publications

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

1. Xintong Liu, Oliver W.W. Yang, "Head-dropping mechanism in traffic control", *Proceedings of SPIE OE 2004*, vol. 5598, pp. 238-245, Philadelphia, PA USA, October 2004.

2. Jun Cong, Xintong Liu, "Design and Analysis of A Traffic Rate Controller", *Proceedings of CCECE 2004*, vol.2, pp. 973-976, Toronto, May 2004.
3. Xintong Liu, Oliver W.W. Yang, "Performance comparison of ECN-BLUE in both single bottleneck and multi-bottleneck networks ", to be presented at IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim2005).
4. Xintong Liu, Oliver W.W. Yang, "Performance evaluation of RED using ECN in a multi-bottleneck network ", under submission to MILCOM2005.
5. Xintong Liu, Oliver W.W. Yang, "AQM techniques in multi-bottleneck networks ", under preparation.
6. Xintong Liu, Oliver W.W. Yang, "Examination of ECN with AQM in bottleneck networks ", under preparation.

Chapter 2 Network Operation, Models and Assumptions

In this chapter, we will first describe the communications network operation mechanism and the network configuration used for our network traffic congestion control studies. Next, the TCP traffic flow models will be presented. Since all simulation work has been done with the OPNET modeler, we will then give an introduction to this powerful network simulation tool associated with the hierarchical models we have developed. Finally, the assumptions made in this thesis will be provided.

2.1 Network Operation

We consider a TCP communications network with a number of nodes, each implementing a traffic congestion control algorithm in its routers. End-to-end applications are considered. A TCP source generates and sends a packet, which contains the source address and the destination address in the IP header of the packet, to the first intermediate router. This router then retrieves the destination address from the IP header of the packet received from the source and forwards the packet to the destination or the next intermediate router if it is not next to the TCP destination. Finally, the router sends the packet to the corresponding TCP destination based on the destination address it obtained from the IP header of the packet if the packet is not dropped. Upon receipt of the packet, the destination generates an ACK and sends it back to the source.

When the source send packets at a transmission rate which exceeds the network's capacity, a queue builds up in the buffer within the router. Once the queue size reaches a threshold (which can be a fraction or the full buffer size), new arrivals are dropped, and a congestion situation is considered occurring. The general congestion control algorithm used in most TCP/IP networks is TCP-Reno. The Reno version of TCP consists of *Slow Start, Congestion Avoidance, Fast Retransmit* and *Fast Recovery* mechanisms. The details can be found in [Jaco90], which has also been summarized in Section 1.2.1.

Two specific network operations are considered in this thesis when congestion occurs, ECN operation and DH operation.

2.1.1 ECN Operation

The main characteristics of the network operation with ECN involvement can be seen in TCP sources, routers and destinations (the source, the router and the destination process models in our implementation). In the TCP source process model, the ECN-enabled source not only responds to the setting of the CE bit but also to normal packet losses (three duplicate ACKs). In the case of packet loss, a source follows the TCP algorithm for congestion control to half its congestion window *cwnd,* reduce the slow start threshold *ssthresh* and retransmit lost packets. Upon receipt of the ACK packet with the CE bit set to 1, the source reacts in the same way as it would with packet loss, but there is no need to retransmit any packet. When an ACK is not marked for congestion, the source follows the TCP algorithm for the normal strategies of traffic control like *Slow Start* and *Congestion Avoidance* to send data and increase the *cwnd* size.

In the router process model, various congestion control techniques can be implemented, and are configurable at the level of network model. Upon the arrival of a packet, if no congestion is detected, the router always queues the packet; if the congestion is detected in the network, the router marks the packet as an indication of congestion and then puts the packet into the queue. The packet is marked with a probability, which is the same probability determined by the original AQM algorithm for dropping packets.

In the TCP destination process model, the ECN-enabled destination checks for congestion notifications in all incoming packets, and forwards the notifications with the ECN-Echo flag set in the next ACK packet to the TCP source. The destination will keep doing this in the subsequent ACKs until it receives an indication from the source that the source has responded to the congestion notification.

2.1.2 DH Operation

The network operation with DH involvement is very similar to the general operation mentioned before. The only difference is that, by operating DH, an incoming packet is inserted at the tail of the queue, and the packet at the head of the queue is discarded, when it is determined that the incoming packet should be dropped.

2.2 Network Configuration

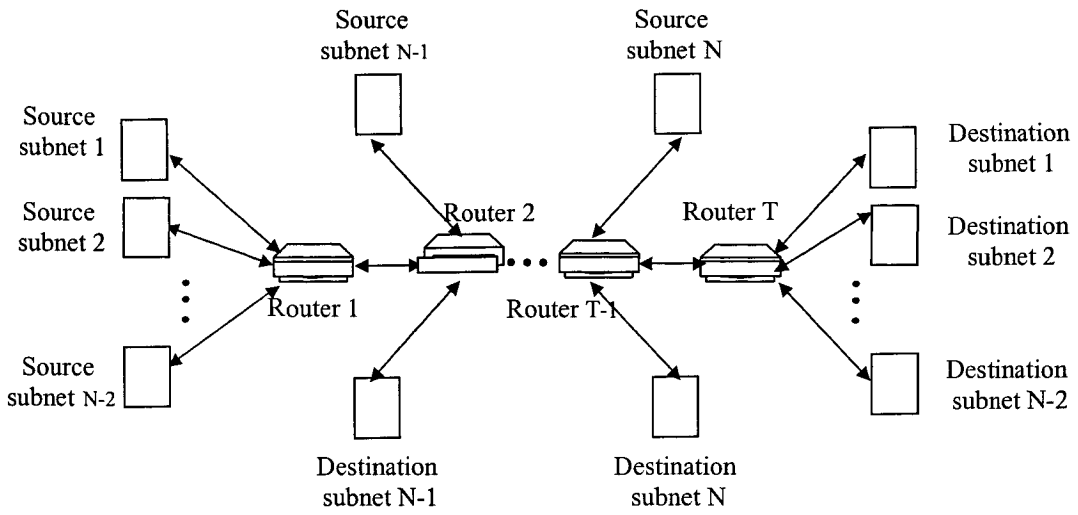

Figure 2-1 General Bottleneck Network Configuration

Figure 2-1 shows a general bottleneck network configuration for our study. It contains N source subnets and N corresponding destination subnets, each of which has M TCP sources/destinations, and T routers implemented with congestion control algorithms. Basically, it can represent the interconnections of LANs (Local Area Network) to WANs (Wide Area Network) or dial-up access users through WANs as in the case of ISP (Internet Service Provider) networks.

For each of the routers, it can be configured with a limited data service rate, e.g. 45 Mbps, or with an infinite data rate. A bottleneck congestion situation may occur at any router if the packet arrival rate exceeds the router data service rate in that router. Thus, this configuration can simulate a single bottleneck or a multi-bottleneck network by setting the rate of a router to a limited rate or not.

2.3 TCP Traffic Flow Models

There are two types of TCP traffic flows used in our study: the short-lived traffic flow and the long-lived traffic flow. Both types of TCP flows have a transmission rate limited by TCP-Reno, a TCP traffic congestion control and avoidance algorithm.

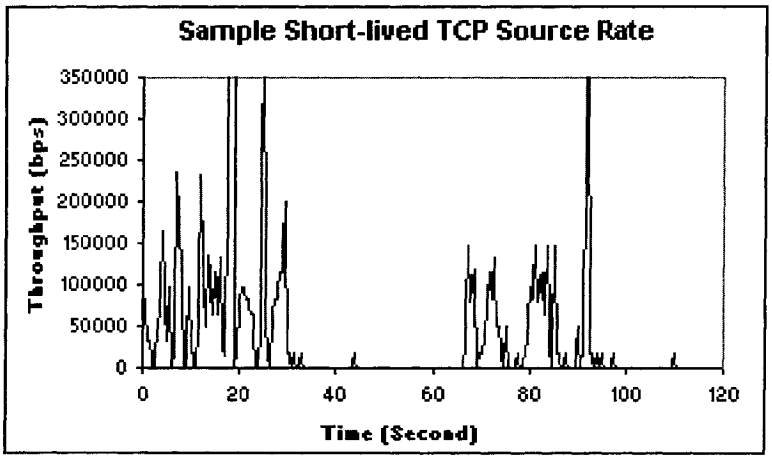

Figure 2-2 Transmission Rate of a Sample of Short-lived TCP Flow

2.3.1 Short-lived TCP Flow

A short-lived TCP flow enters a network after an undetermined think time, generates and sends a file of an undetermined but limited length, and then gets into an idle period for another think time period after one file transfer completes. HTTP (Hypertext Transfer Protocol) source is considered as a typical example of the short-lived source.

Researchers have done much study on the traffic pattern of the short-lived TCP flow [CrBe97], and proposed various probability functions to describe the characteristics of the distributions of file length and think time, such as the self-similarity model [LeTa94] and wide area traffic model [CrBe97]. In our study, we have adopted the mathematical model proposed by Ott et al. [OtLa99]. This model simulates the characteristic of file length (in bytes) by using the probability distribution of Equation (2-1), which has a finite mean and an indefinite variance:

$$P\{F > f\} = \left[1 + (\frac{f}{\pi})^{\alpha}\right]^{-1} , \tag{2-1}$$

where π is the median and α is a positive constant. The think time (in seconds) between any two transfers is modeled as the probability distribution of Equation (2-2)

$$P\{T > t\} = p_h\left[1 + (\frac{t}{\pi_h})^{\eta_h}\right]^{-1} + p_l\left[1 + (\frac{t}{\pi_l})^{\eta_l}\right]^{-1} , \tag{2-2}$$

where p_h, p_l, π_h, π_l, η_h and η_l are constant parameters, and $p_h + p_l = 1$. The distribution of Equation (2-2) is the mixture of two distributions of Equation (2-1). It has been shown that the traffic pattern of short-lived TCP flows controlled by these two distributions is realistic [OtLa99].

The transmission rate of a short-lived TCP source is also limited by the real-time bandwidth available in the network. Figure 2-2 shows the transmission pattern of HTTP sources, under the congestion controls of TCP-Reno and RED.

2.3.2 Long-lived TCP Flow

A long-lived TCP source, unlike the short-lived TCP source, always has data to send (infinitely long file length) as long as it is allowed to do so by the TCP congestion window size. The greedy FTP (File Transfer Protocol) source is an example of the long-lived TCP source. A long-lived TCP source can also be modeled by the think time equation mentioned above, taking zero value for the think time to make sure its file size is infinitely long.

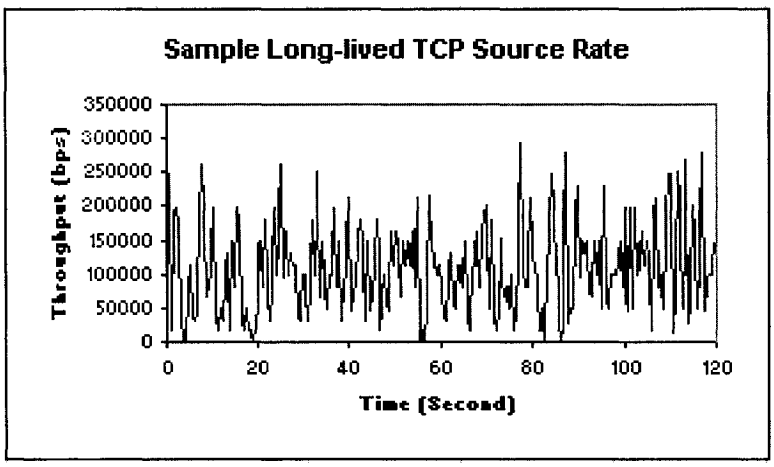

Figure 2-3 Transmission Rate of a Sample of Long-lived TCP Flow

Like the short-lived TCP flow, the transmission rate of a long-lived TCP flow is also limited by the real-time bandwidth available in the router. Figure 2-3 shows the transmission pattern of FTP sources, under the congestion controls of TCP-Reno and RED.

2.4 OPNET Models

OPNET [OPNE00] is an engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis. Features include the graphical specification of models, the dynamic and event-scheduled simulation kernel, the integrated data analysis tools and hierarchical and object-based modeling.

A hierarchical OPNET model is composed of network models, node models and process models. The top level is the network model made of individual nodes, including TCP sources, TCP destinations, routers and links connecting the sources, the destinations and the routers. The node models are under the network model level. Each node model reflects the behaviour of its correspondingly individual node in the network model, and consists of modules, each of which has its own process model, forming the process model level. A process model is a Finite State Machine (FSM), which represents a module's behavior, and is made up of any number of process states that a module may be in, and the necessary criteria for changing states in response to different events. Each state can be programmed using a C like language called Proto-C. These codes, as the heart of the OPNET models, deal with concrete and specific functions.

We have upgraded OPNET TCP model package by upgrading the original single bottleneck network model and extending this upgraded single bottleneck network model to a multi-bottleneck one, both of which will be introduced in this section. Verifications for the correctness of these models will also be discussed.

There are some useful features in our upgraded OPNET TCP model when compared with its predecessor (OPNET TCP model package), which are summarized as follows:

1. Almost all simulation parameters in the upgraded OPNET TCP model are configurable in the level of network model, while in the original OPNET TCP model, the simulation parameters are accessible in the simulation sequence and other different levels of the model. The former makes the configuration of simulation attributes more convenient.
2. In the package of the upgraded OPNET TCP model, all similar process models have been incorporated into only one process model. For example, we have combined the process models where all routers use only one data and ACK process model pair, in which the system can automatically distinguish different routers according to the setting of *router_id*, with the multiplexer and the demultiplexer process models now shared by

both source subnets and destination subnets. This characteristic makes process model management and maintenance simpler. While in the OPNET TCP model set, each node has its own process model even though most of these process models are the same or very similar to each other.

3. We have provided the integrated long-lived and short-lived TCP flow model in the upgraded OPNET TCP model set, where either one or both of them can be easily chosen by users at the network level, while there are two separate flow models in the OPNET TCP model. Likewise, this improved feature is for the purpose of simplification.
4. Congestion control algorithms are more easily to be implemented in our upgraded model when compared with the old one because it indicates where the code of the algorithm can be written.

2.4.1 Single Bottleneck Network Model

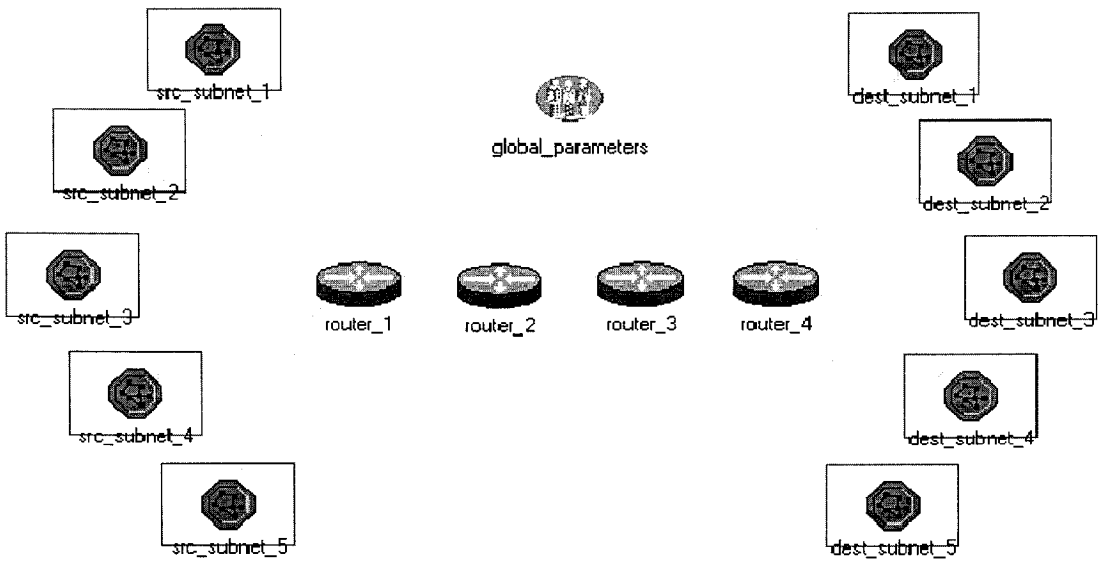

Figure 2-4 OPNET Single Bottleneck Network Model

Based on the general bottleneck network configuration shown in Figure 2-1, we built our single bottleneck network model, as shown in Figure 2-4. It can be seen that this figure is similar to Figure 2-1, with the same intuitive description of the network. This single bottleneck network model contains 5 source subnets and 5 destination subnets (N is set as 5) with 100 (M is set as 100) sources/destinations for each subnet, and 4 routers. We configure

the data service rate of Router 1 (the bottleneck router) as 45 Mbps and the others as infinity. The node "global_parameters" in the figure keeps some predefined simulation parameters such as the upper bound window size for a TCP source, which was set to be 10000 packets, and the global segment size, which was set to 1024 bytes.

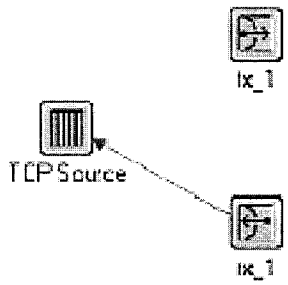

Figure 2-5 OPNET TCP Source Node Model

Figure 2-5 shows the OPNET TCP source node model. There are three different objects in the figure. The TCP source generates packets according to the particular TCP mechanism (e.g., TCP-Reno), and sends them to the transmitter, an OPNET built-in object. The transmitter then forwards these packets to the network link. The receiver, also an OPNET built-in object, receives ACKs from the destination, which are transmitted to the TCP source through the network link. The TCP source then processes these ACKs and responds accordingly.

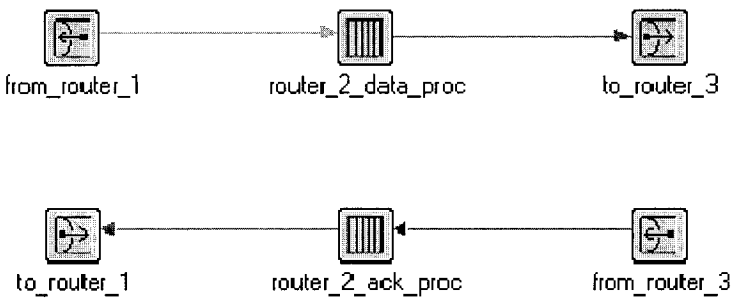

Figure 2-6 OPNET Router Node Model (for Router 2) in Single Bottleneck Network

Other OPNET node models, such as the TCP destination node model and the router node model, have the same mechanisms. As an example, we show Router 2's node model in Figure 2-6, which will be compared with that in our multi-bottleneck network model later.

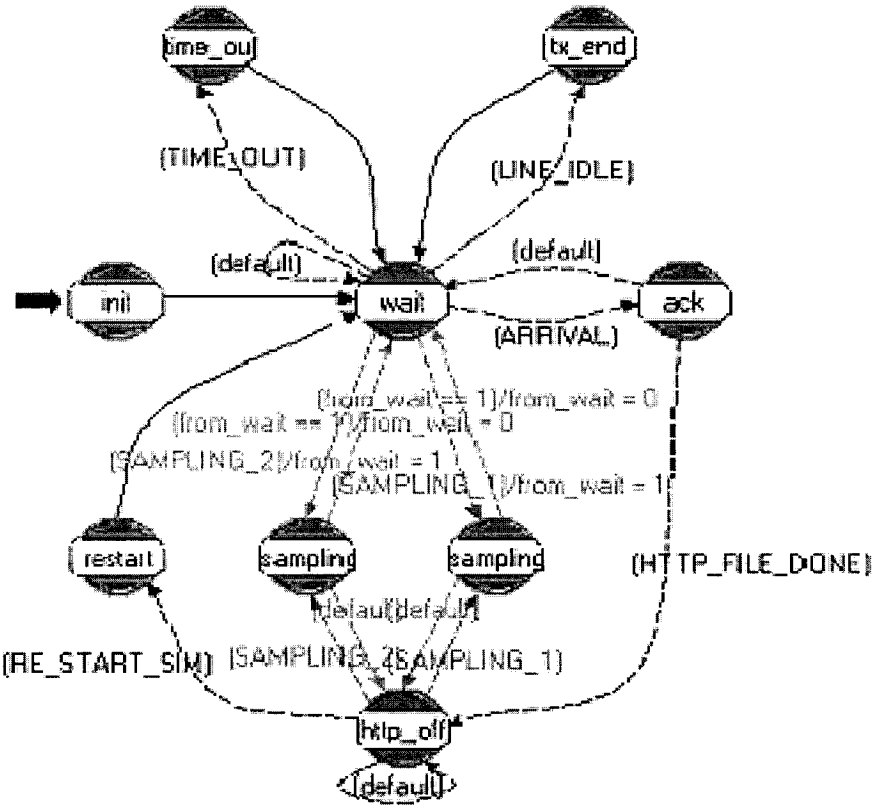

Figure 2-7 OPNET TCP Source Process Model

Figure 2-7 shows the OPNET process model for the integrated long-lived and short-lived TCP source. The process model includes all possible states of a TCP source while under the control of TCP-Reno. To initialize the source parameters in a simulation, the system is in the *init* state when the simulation begins. Then it goes into the *wait* state automatically. Upon receipt of an ACK, the TCP source enters the *ack* state to respond accordingly. The TCP source enters the *time_out* state to send lost packets and reschedule the next timeout when the timeout event occurs, and enters the *tx_end* state to send out TCP packets when the link has been idle. The *sampling_1* and *sampling 2* states are to collect statistic at fixed time intervals for later data analysis. Note that the *HTTP_off* state and the *restart* state can only be stepped into when the TCP source is a short-lived source.

Figure 2-8 shows the OPNET router data process model, which exhibits how the router works when a packet arrives. This model is built based on the M/M/1 queuing model, an integrated model in the OPNET modeler. It has five main states, i.e. the *init* state, the *arrival* state, the *svc_star* state, the *svc_compl* state and the *idle* state. When a simulation starts, the

system enters the *init* state to initialize the simulation parameters, and then enters the *idle* state. Upon the arrival of a packet, the router enters the *arrival* state where our studied congestion control algorithms (ECN-RED, ECN-BLUE, ECN-ARED, ECN-PI-RED, DH-RED, DH-BLUE, RED, BLUE, ARED, and PI-RED), are implemented. Then it goes into the *svc_start* state to get served if the server is not busy and the packet is successfully inserted in the router queue. Otherwise, it enters the *idle* state. During the *idle* state, the router goes into the *svc_compl* state if the packet service is finished. Then, if the queue is empty, it enters the *svc_start* state. Otherwise, it enters the *idle* state. The *sampling_1* and the *sampling 2* states are to obtain the statistic sampled at a fixed time period for later data analysis. Especially, the *ARED_sampling* state and the *PI_sampling* state are created for collecting the statistics of the ARED and PI-RED related algorithms, respectively.

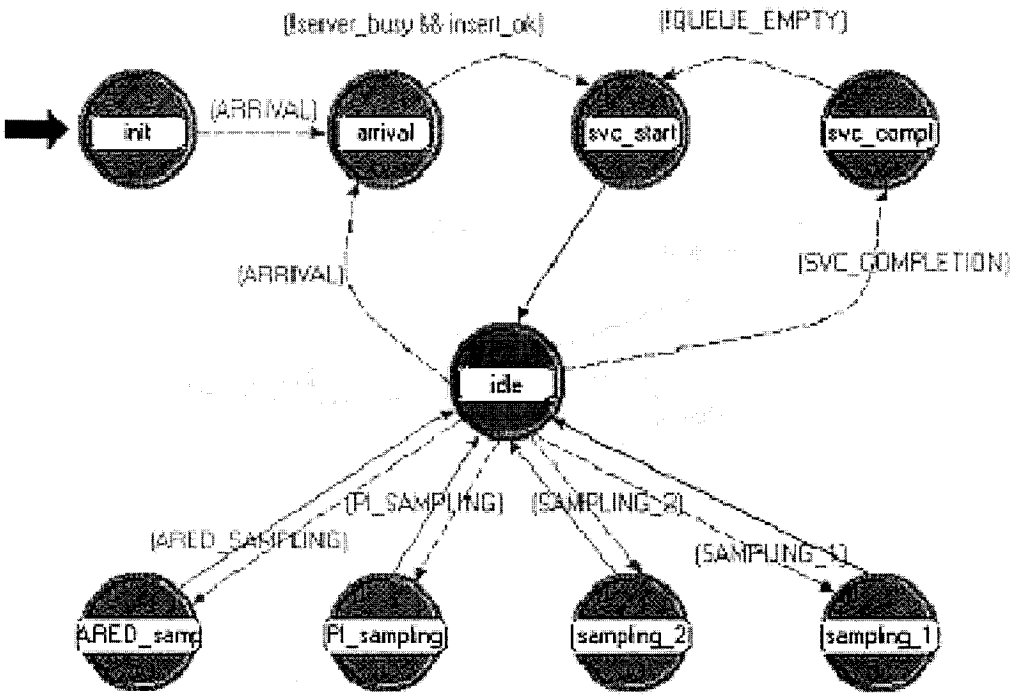

Figure 2-8 OPNET Router Data Process Model

Except for the TCP source process model and the router process model, we still use some other OPNET process models in this thesis, such as the source subnet multiplexer process model, the source subnet demultiplexer process model, the router ACK process model, the TCP destination process model, the destination subnet multiplexer process model,

the destination subnet demultiplexer process model, and the *global_parameters* process model.

In addition, we build some other OPNET models, including the packet format model, the link model, the probe model and the simulation sequence, all of which are necessary parts to form the whole OPNET simulation model, for serving our simulation-based study.

2.4.1.1 Verification of the Single Bottleneck Network Model

To verify the correctness of our upgraded single bottleneck network model with respect to the old single bottleneck network model, which has been validated by its users, we compare the performances of a certain algorithm (RED) in both of the models, in terms of average queue size and average packet drop rate, under the same simulation environment.

The simulation environments we set for both of the models are the same, which are summarized as follows: all network systems have the simulated time of 120 seconds and the average packet size of 1024 bytes. We set the data service rate of Router 1(the bottleneck router) as T3 (45 Mbps) while setting the others as infinity, in order to make sure there is only one bottleneck router in the network. The router buffer size is set to 600 packets. We use 100 FTP sources. The end-to-end round trip propagation delay is set to 0.1 seconds. The parameters of RED are configured based on the recommended values in the original paper, that is, the maximum packet drop probability is 0.1, the minimum queue threshold is 150 packets, the maximum queue threshold is 400 packets, and the queue weight is 0.002. Besides, we ensure the other configurations such as the initial timeout time, which is set to 0.5 seconds, and the method of statistic collection, which is by using the node attributes incorporated in OPNET, be the same in both of the models.

We have run quite a few simulations in both of the models, and obtained the similar simulation results. In this subsection, we present Figures 2-9 and 2-10 as our examples of those results, which show the average queue size and the average packet drop rate comparisons of RED in the old model with RED in the upgraded model, respectively. We can see the similar performance of RED in both of the models. Therefore the correctness of the upgraded single bottleneck model can be verified.

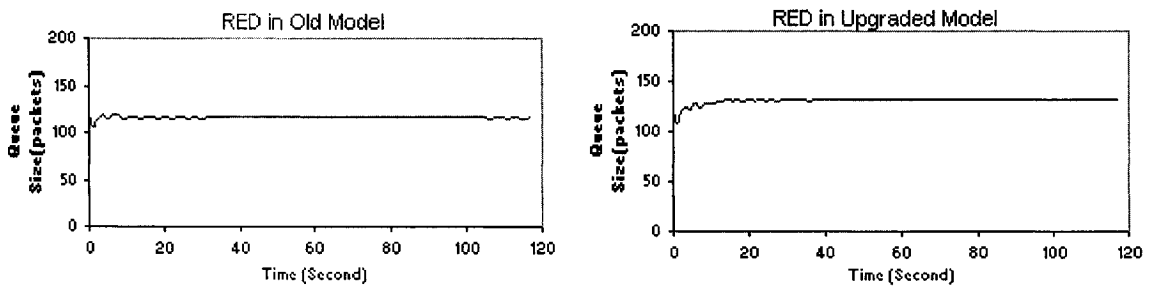

Figure 2-9 Average Queue Size Comparison with 100 FTP Sources

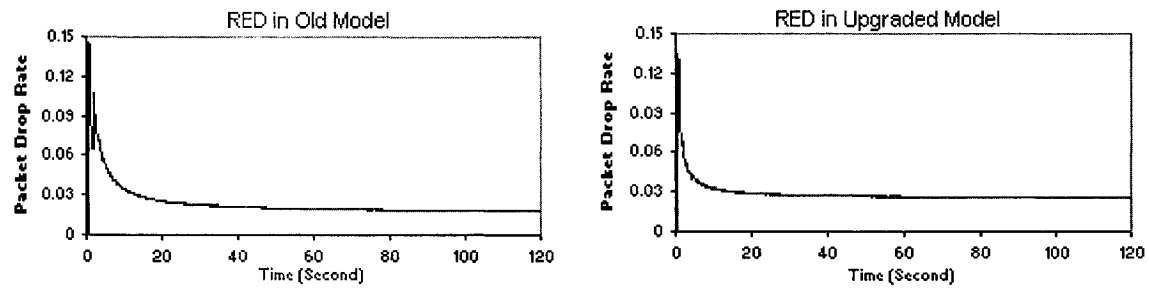

Figure 2-10 Average Packet Drop Rate Comparison with 100 FTP Sources

2.4.2 Multi-bottleneck Network Model

Figure 2-11 shows the multi-bottleneck network model we built based on the single bottleneck network shown in Figure 2-4. It consists of 7 source subnets and 7 destination subnets with 100 TCP sources/destinations for each subnet (700 sources and 700 corresponding destinations in total), and 4 routers, in which our congestion control algorithms are implemented. By setting all of the four routers in Figure 2-11 to have limited data service rates (e.g. all four routers can be set to the same data rate of 45 Mbps, referred to "45-45-45-45 Network" in this thesis), we make them behave as the bottleneck routers in our multi-bottleneck network model. The traffic flows generated from different source subnets pass through the different paths to the corresponding destinations as indicated in Table 2-1. Even though each of the links in Figure 2-11 is configured with the same propagation delay of 0.01 seconds, the RTTs (which consists of propagation delay and queuing delay) for the traffic from different source subnets are actually different regardless whether their queuing delays are different since the total propagation delays are different due to the different paths they pass. For example, the RTTs for the traffic respectively from Source subnets 1, 4 and 5

are different from each other. Like in the single bottleneck network model, some predefined simulation parameters are kept in the node of "global_parameters".

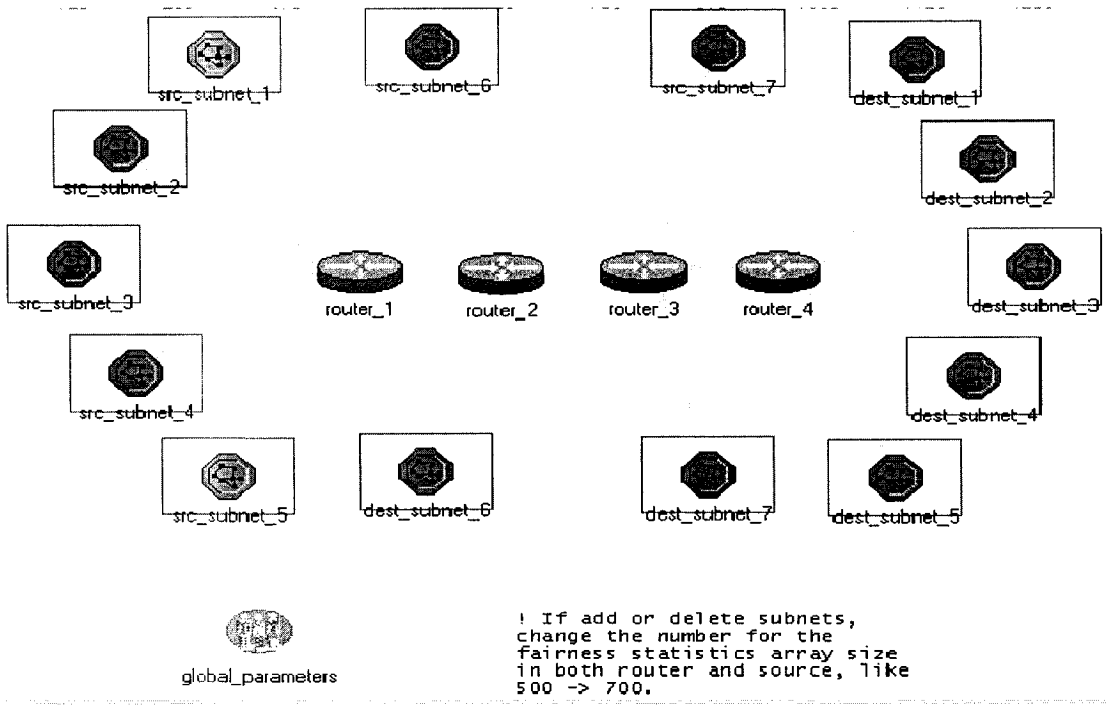

Figure 2-11 OPNET Multi-bottleneck Network Model

Table 2-1 Different Paths (Source-router-destination) and Round Trip Propagation Delays for Traffic from Different Source Subnets

Source subnet	Router passed	Destination subnet	Total round trip propagation delay
Source subnet 1	Routers 1, 2, 3, 4	Destination subnet 1	0.14 second
Source subnet 2	Routers 1, 2, 3, 4	Destination subnet 2	0.14 second
Source subnet 3	Routers 1, 2, 3, 4	Destination subnet 3	0.14 second
Source subnet 4	Routers 1, 2, 3	Destination subnet 7	0.12 second
Source subnet 5	Routers 1, 2	Destination subnet 6	0.10 second
Source subnet 6	Routers 2, 3, 4	Destination subnet 5	0.12 second
Source subnet 7	Routers 3, 4	Destination subnet 4	0.10 second

Except for the different node models for Routers 2 and 3 in the multi-bottleneck network (as shown in Figures 2-12 and 2-13, respectively), most OPNET models in the multi-bottleneck network have the same physical topologies and the basic mechanisms as those in the single bottleneck network. These OPNET models are: the source subnet network model, the destination subnet network model, the source node model, the destination node model, the node models for Routers 1 and 4, the source process model, the destination

process model, the router data process model, the router ACK process model, the source subnet multiplexer process model, the source subnet demultiplexer process model, the destination subnet multiplexer process model, the destination subnet demultiplexer process model, and global_parameters process model. However, the internal codes under these models are different from those in the single bottleneck network. For example, the code in the multi-bottleneck network reflects more complicated logical relationship on the packet delivering within the routers, the sources, and the destinations, which also forms the main difference between these two network models. Likewise, we have the similar packet format models, the similar link models, the similar probe models, and the similar simulation sequences for both of the single bottleneck network and the multi-bottleneck networks.

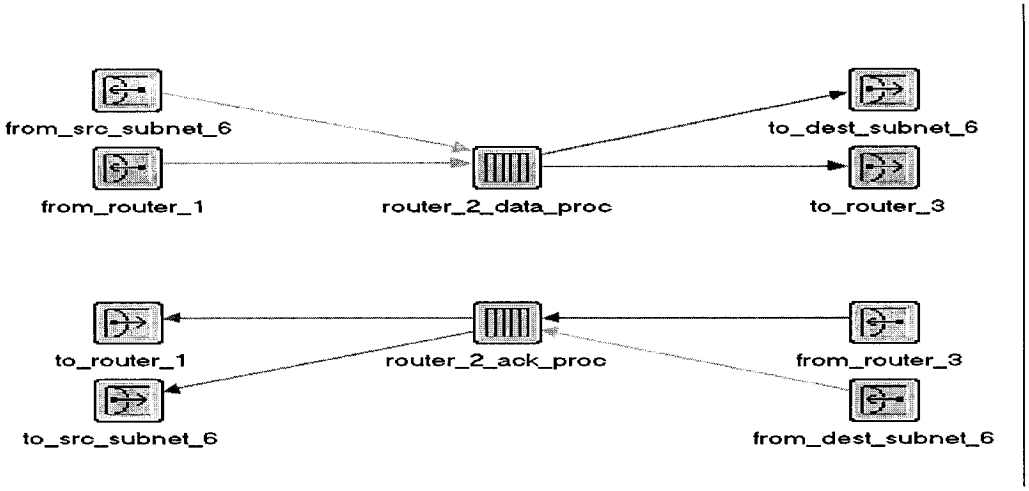

Figure 2-12 OPNET Router Node Model (for Router 2) in Multi-bottleneck Network

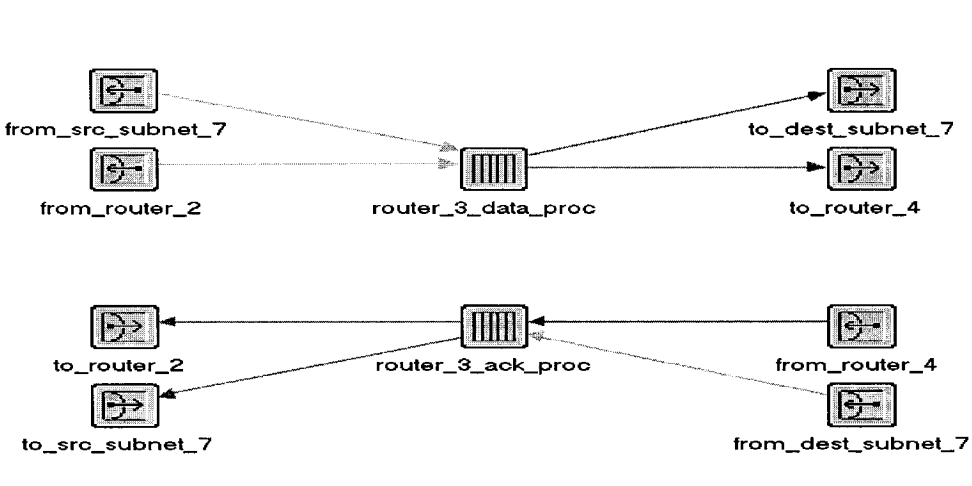

Figure 2-13 OPNET Router Node Model (for Router 3) in Multi-bottleneck Network

2.4.2.1 Verification of the Multi-bottleneck Network Model

Likewise, the correctness of our multi-bottleneck network model can be verified by comparing the performances of RED in the multi-bottleneck network model with those in the upgraded single bottleneck model, which has been verified in Section 2.4.1.1, in terms of average queue size, and average packet drop rate, under the same simulation environment.

The simulation environments we created for both of the network models are the same: all network systems have the simulated time of 120 seconds and the average packet size of 1024 bytes. We set the data service rate of Router 1 to T3 while setting the others to infinity. The router buffer size is set to 600 packets. We use 300 FTP sources. The end-to-end round trip propagation delay is set to 0.1 seconds. The RED parameters used for the verification are the same as those in Section 2.4.1.1.

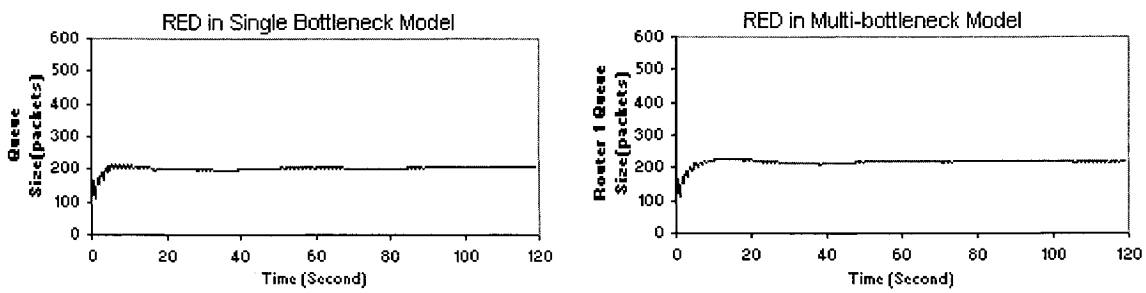

Figure 2-14 Average Queue Size Comparison with 300 FTP Sources

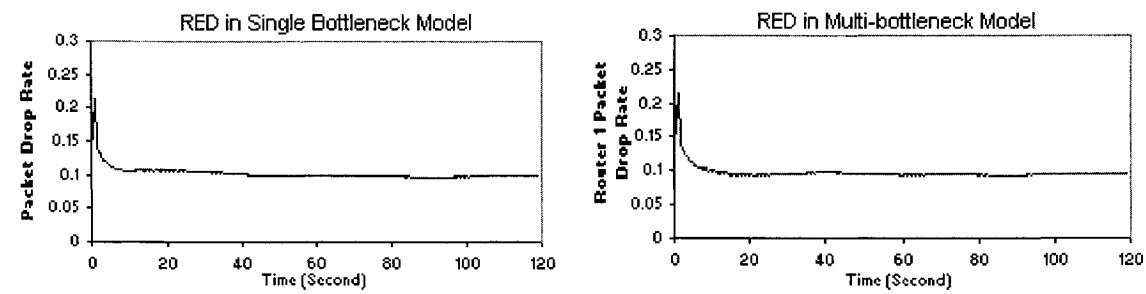

Figure 2-15 Average Packet Drop Rate Comparison with 300 FTP Sources

As with the verification of the upgraded single bottleneck network model we have done in Subsection 2.4.1.1, we have also run quite a few simulations in both of the models, and achieved the similar simulation results. Here, we show Figures 2-14 and 2-15 as our examples of those results, which show the average queue size and the average packet drop rate comparisons of RED in the single bottleneck network model with RED in the multi-

bottleneck network model. The similar performances of RED verify the correctness of the multi-bottleneck network model.

2.5 Assumptions

For this section and the remainder of the thesis, the following assumptions pertain:

1. Each source/destination pair is one to one mapping and does not have cross traffic. This would allow us to simplify the simulation and evaluation of the queuing performance in the bottleneck router.
2. The TCP destination does not send any packets actively, unless it has received a packet from the source, and in that case an ACK is sent back to the source. This allows us to simplify the simulation and limit the effect of any factors other than those in the congestion control algorithms used in our performance comparison.
3. The destination's receiver window size is set large enough so that TCP connections are not constrained there. This would allow us to focus on the queuing performance evaluation of the congestion control algorithms residing in the router only.
4. The traditional TCP flows (i.e., long-lived FTP sources and short-lived HTTP sources) are used in our simulation networks. We do not consider multimedia traffic.
5. All of the FTP and HTTP sources are sending packets at the same time (in the beginning of the simulation).
6. All packets used in our simulations have the same size.

Chapter 3 ECN with AQM in a Single Bottleneck Network

We will first discuss the features of various ECN-AQM schemes, specifically, ECN-RED, ECN-BLUE, ECN-ARED and ECN-PI-RED. Then we will evaluate the performance of these algorithms through simulations and comparisons with their corresponding AQM algorithms in the single bottleneck network.

3.1 ECN with AQM

We present in the following the specific procedures of four algorithms but mainly focus on the part related to ECN. The detailed descriptions of the original AQM algorithms we use will be summarized in Appendix A.

3.1.1 ECN-RED

Our ECN-RED algorithm uses a low-pass filter with an exponential weighted moving average to calculate the average queue size avg . Like RED, this is then compared to two preset parameters, the minimum queue threshold $th_{\min}$ and the maximum threshold $th_{\max}$.

1) If $avg < th_{\min}$, then the router always queues incoming packets (no packet has to be dropped or marked).
2) If $avg > th_{\max}$, then the router marks every incoming packet and queues it at the tail of the queue.
3) If $th_{\min} \leq avg \leq th_{\max}$, then the router marks an incoming packet, instead of dropping the packet as in RED, with a probability, which is also used by RED for packet dropping, and then queues it at the tail of the queue. However, when it is determined that no marking is required, the packet is inserted at the tail of the queue.

3.1.2 ECN-BLUE

If the average queue size is less than the buffer size, an incoming packet is marked by ECN-BLUE, and then queued, instead of being dropped as in BLUE. The packet marking probability is the same probability used by BLUE for its packet dropping. However, if it is determined that no marking is required, the packet is inserted at the tail of the queue. Moreover, this marking probability, like in BLUE, will be updated by decreasing by a constant factor after the link remains idle for a given period of time (called freeze time, which is used to determine the minimum time interval for updating the packet-dropping probability), and by increasing by another constant factor after the queue has been discarding packets due to buffer overflow for a period of the freeze time.

3.1.3 ECN-ARED

If the average queue size is between the maximum queue size threshold and the minimum queue size threshold, an incoming packet is marked instead of being dropped, and then queued. The packet marking probability in ECN-ARED is the same one as that used by ARED for its packet dropping. Likewise, when it is determined that no marking is required, the packet is inserted at the tail of the queue. On the other hand, any incoming packet is discarded (even though ECN is involved) if the average queue size is larger than the maximum threshold, and is always queued if the queue size is less than the minimum threshold. We have noticed that the difference between ECN-ARED and ECN-RED is the difference between the original ARED and RED algorithm.

3.1.4 ECN-PI-RED

If the average queue size is less than the buffer size, an incoming packet is marked by ECN-PI-RED with the same probability as that used by PI-RED for packet dropping, which is computed by a PI controller. For each probability sampling time, if there is a difference between the average queue size and the reference queue size, then the PI controller will verify its output to change the packet drop probability accordingly. However, when it is determined that no marking is required, the packet is inserted at the tail of the queue.

In addition, the source in ECN-AQM needs not retransmit the packets when responding to marked packets, even though it has to retransmit lost packets when responding to normal

packet losses. Also note that the above procedures are the actions taken by the algorithms only when the average queue size is less than the buffer size. Otherwise, all incoming packets will be discarded from the tail of the queue even though we apply ECN to AQM.

As can be seen from the above subsections, each pair from the two different groups of schemes (e.g. ECN-RED from the group of the ECN-involved AQM schemes, and RED from the corresponding group of the non-ECN-involved AQM schemes) shares the same core technique, which is the computational method of the packet drop/mark probability. The only difference between them is an incoming packet is marked, instead of being dropped when ECN is involved and the conditions for marking packets are satisfied according to the different ECN-AQM algorithms. These conditions are the same conditions, according to which AQM algorithms decide whether discarding packets or not.

3.2 Simulation Studies

We use the single bottleneck network model, as shown in Figure 2-4 and described in Section 2.4.1, to do simulations and to evaluate performances. All network systems are simulated for a time period of 120 seconds (which we have determined to be adequate to obtain reasonable steady-state results), with an average of 5500 packets per second served in the bottleneck router when the router data service rate was set as T3 (45 Mbps), and all of the packets are 1024 bytes (45000000/1024x8=5500 packets).

In our simulation model, the sources are implemented with TCP-Reno. In the TCP source model, the parameters for the file length probability distribution function of Equation (2-1) are $\pi = 2190$ and $\alpha = 1.15066$, and the parameters used in the think time probability distribution function of Equation (2-2) are $p_h = 0.4953916$, $p_l = 0.5046084$, $\pi_h = 1.0$, $\pi_l = 0.0245032$, $\eta_h = 1.243437$ and $\eta_l = 3.252665$.

Each of the links in Figure 2-4 is configured with the same propagation delay of 0.01 seconds, which means the round trip end-to-end propagation delay is 0.01x7x2=0.14 seconds. However, considering random queuing delay, the RTT for each end-to-end connection in the network can be different from each other.

The main parameters of all of the investigated algorithms used for our simulations are listed in Table 3-1. Besides, the buffer size for each of the routers is set to 600 packets in all scenarios. We set most of these parameters according to the recommended values in the

original papers [FlJa93, FeKa02, FlGu01, HoMi01b]. It is reasonable to use such values to ensure a fair comparison. All of the parameters are configurable at the OPNET network model level.

Table 3-1 Simulation Parameters

RED, RED-ECN	
Minimum queue threshold ($th_{\min}$)	150
Maximum queue threshold ($th_{\max}$)	400
Maximum packet drop probability ($p_{\max}$)	0.1
Queue averaging filtering gain (w_q)	0.002
BLUE, BLUE-ECN	
Initial packet drop probability (*p(0)*)	0.05
Minimum time interval for updating the drop probability (*freeze_time*)	0.01
Increase drop probability (*d1*)	0.00025
Decrease drop probability (*d2*)	0.000025
ARED, ARED-ECN	
Minimum queue threshold ($th_{\min}$)	150
Maximum queue threshold ($th_{\max}$)	400
Maximum packet drop probability ($p_{\max}$)	0.1
Queue averaging filtering gain (w_q)	0.002
Sampling interval (Δt)	0.5
Increment parameter for maximum packet drop probability (α_{ARED})	0.01
Decrement parameter for maximum packet drop probability (β_{ARED})	0.9
Queue size control target (*T*)	275
PI-RED, PI-RED-ECN	
Reference queue size (q_0)	100
Round-trip time limit (R_0)	0.1
Traffic load factor (*N*)	500
Sample frequency (1/*Sampling_interval*)	100

Based on the single bottleneck network model, we create three simulation scenarios, each with different number of FTP sources and HTTP sources, as shown in Table 3-2, in order to see how different types of TCP sources affect the performance of these algorithms.

Table 3-2 Numbers of FTP and HTTP Sources in Different Simulation Scenarios for Single Bottleneck Network

Scenario No.	Number of FTP sources	Number of HTTP sources	Network Configuration
1	500	0	Single Bottleneck Network
2	300	200	Single Bottleneck Network
3	100	400	Single Bottleneck Network

Table 3-2 shows the number of FTP and HTTP sources in the different simulation scenarios for our single bottleneck network. All of the 500 TCP sources are FTP sources in Scenario 1. There are 300 FTP sources and 200 HTTP sources in Scenario 2, and 100 FTP sources and 400 HTTP sources in Scenario 3.

3.3 Performance Evaluation

In this section, we will compare ECN-RED, ECN-BLUE, ECN-ARED and ECN-PI-RED, with RED, BLUE, ARED and PI-RED, respectively, in three single bottleneck network scenarios.

There are two performance metrics used for our comparison. The queue size is defined as the number of pending packets sampled at fixed intervals of 1 second in the bottleneck router queue. Queue size is an important performance measurement for a TCP network. Most AQM schemes depend on the queue size to detect early congestion. We will present the instantaneous queue size performance in order to observe the queue size stability. Another metric is the average packet drop rate, which is defined as the number of discarded packets divided by the time duration between two consecutive sampling instances in the bottleneck router. Packet dropping results in retransmission, which wastes network resources, lowers efficiency, and sometimes makes congestion situation even worse. All of these measurements are taken from the bottleneck router (Router 1).

3.3.1 Scenario 1 (500 FTP sources and no HTTP source)

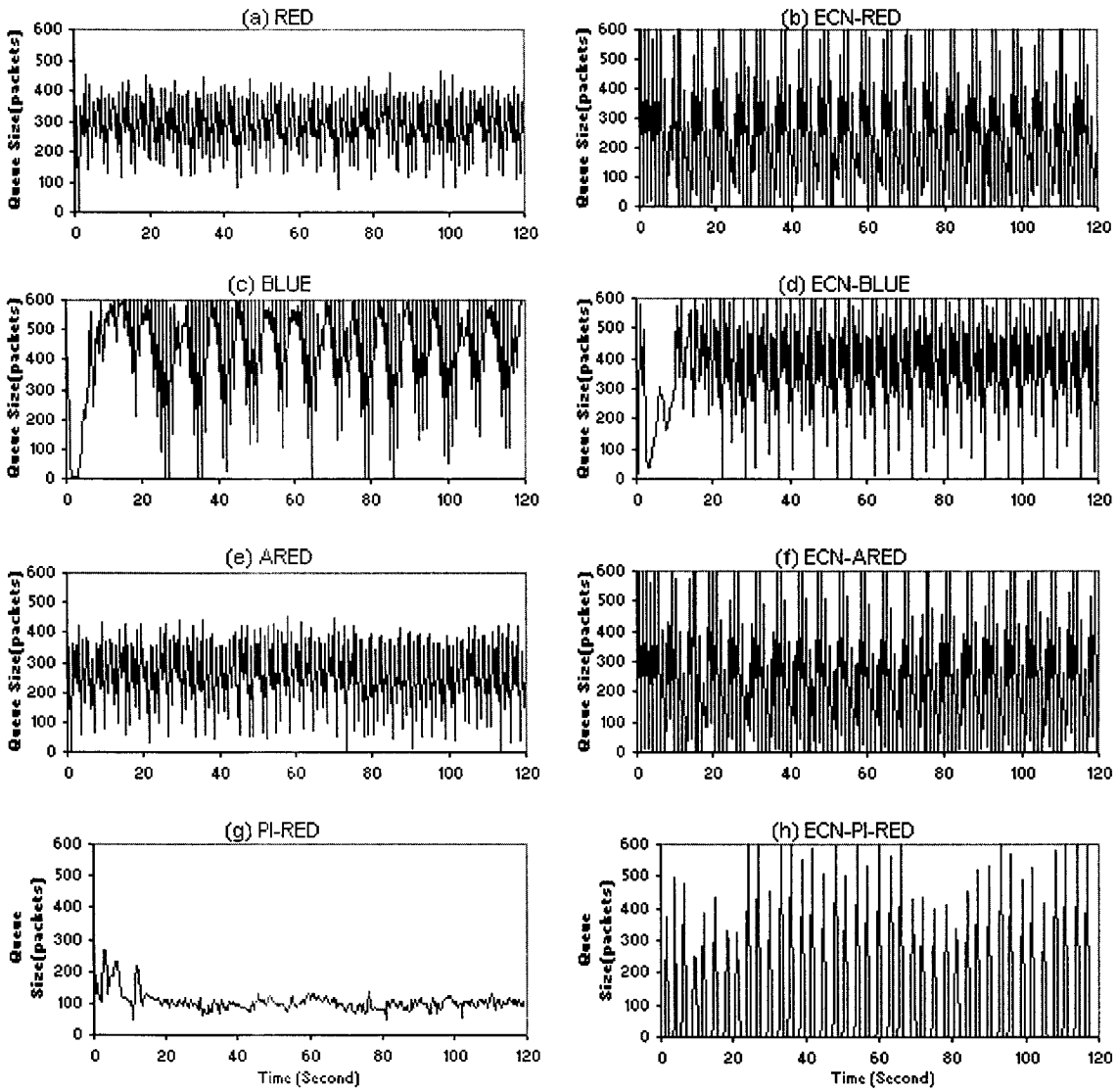

Figure 3-1 Instantaneous Queue Size Comparison of ECN-AQM with AQM for Scenario of 500-FTP in Single Bottleneck Network

Figure 3-1 shows the instantaneous queue size comparisons of ECN-RED with RED, ECN-BLUE with BLUE, ECN-ARED with ARED, and ECN-PI-RED with PI-RED in Scenario 1 (500 FTP sources). It can be seen from Figure 3-1 that ECN-AQM schemes such as ECN-RED and ECN-ARED generally present larger queue size fluctuations than the corresponding AQM algorithms. In particular, ECN-PI-RED presents the extremely severe queue size fluctuation when compared with PI-RED because PI-RED has a strong ability to control its queue size at a reference level, based on its algorithm, and this advantage is lost when ECN is

involved. Also, we notice that ECN-BLUE has severe queue size oscillation, which is similar to BLUE, because BLUE performs poorly in its queue size stability according to its inherent features, and the application of ECN to BLUE does not help it.

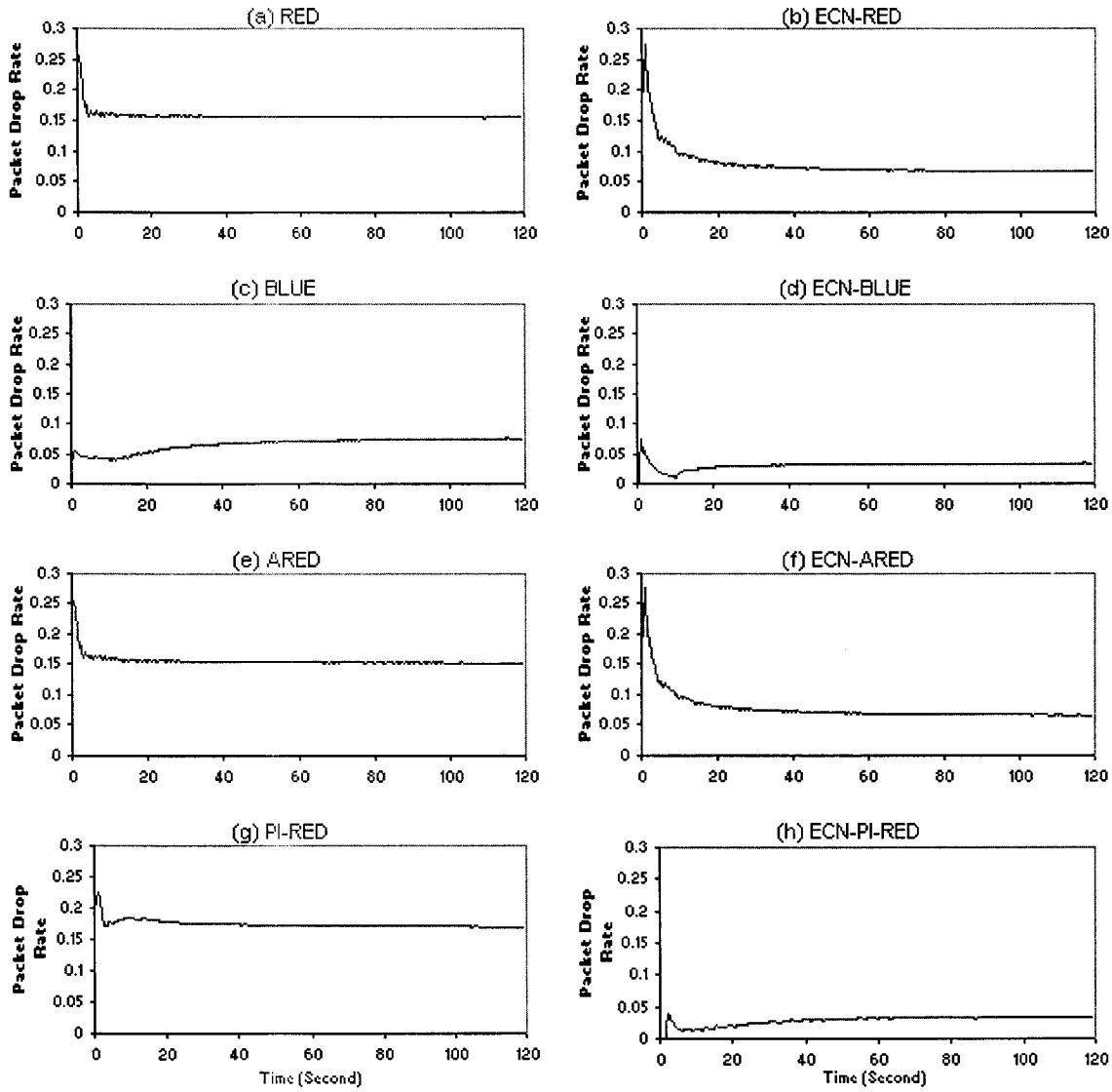

Figure 3-2 Average Packet Drop Rate Comparison of ECN-AQM with AQM for Scenario of 500-FTP in Single Bottleneck Network

Figure 3-2 shows the average packet drop rate comparisons of those eight algorithms in Scenario 1. It is not surprising that all of the investigated ECN-AQM algorithms have lower packet loss rates than the corresponding AQM algorithms. The lower packet drop rate of ECN-AQM comes from the effort made by the ECN mechanism, which marks packets instead of dropping them. This is especially true for ECN-PI-RED since it exhibits an

extremely lower packet drop rate than PI-RED because the application of ECN to PI-RED has a strong effect on decreasing the packet drop rate of the original PI-RED algorithm.

3.3.2 Scenario 2 (300 FTP source and 200 HTTP sources)

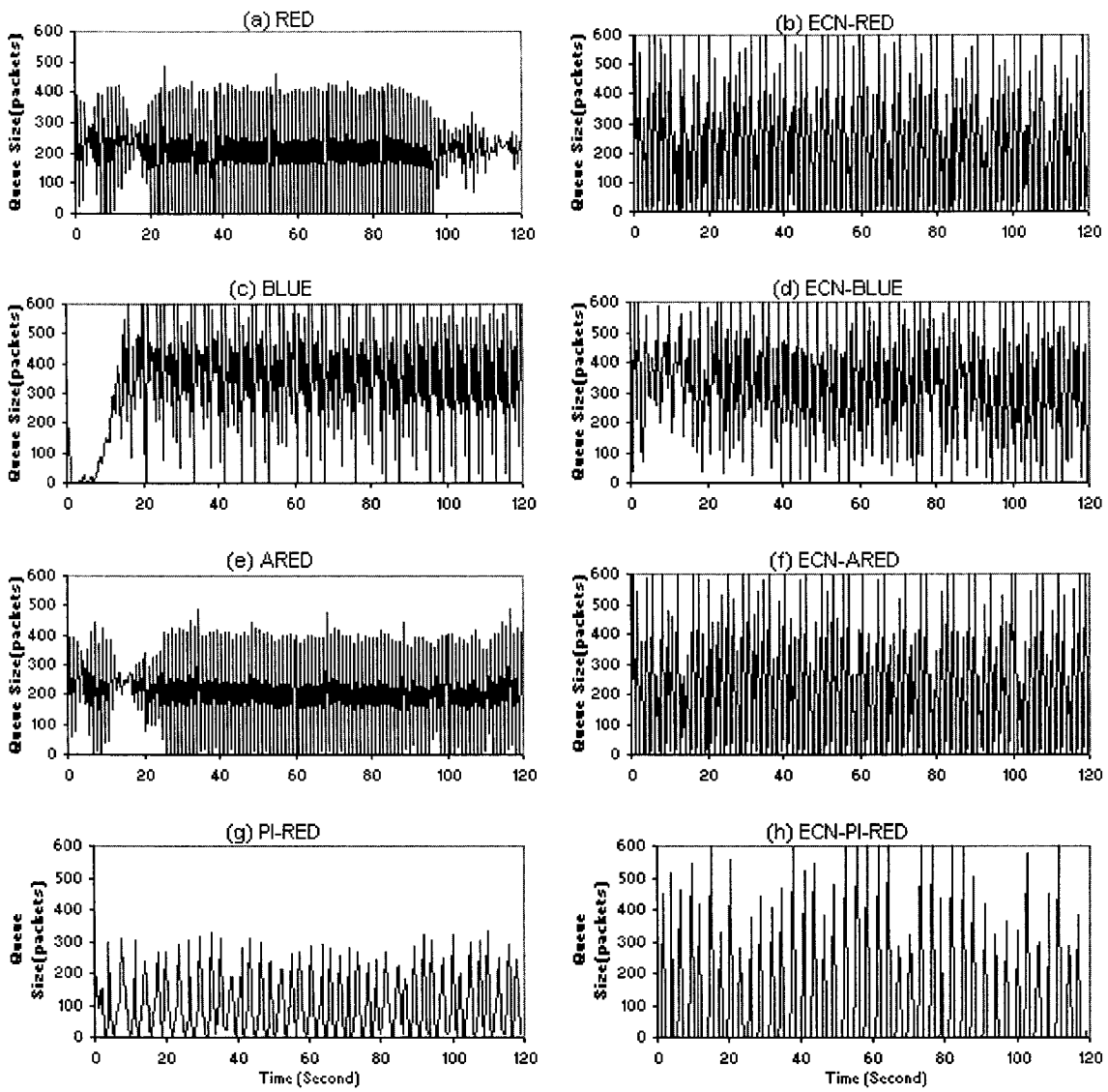

Figure 3-3 Instantaneous Queue Size Comparison of ECN-AQM with AQM for Scenario of 300-FTP-plus-200-HTTP sources in Single Bottleneck Network

The instantaneous queue size comparisons of all those eight algorithms in Scenario 2 are shown in Figure 3-3. As can be seen, even though each of the investigated algorithms behaves differently in Scenario 2 when compared with the same algorithms in Scenario 1 (e.g. AQM schemes in Scenario 2 generally present greater queue size oscillation than in

Scenario 1), the relationship between ECN-AQM and AQM is similar for both scenarios. That is, ECN-AQM generally shows larger queue size fluctuations than AQM, except for ECN-BLUE and BLUE, for the same reasons stated in Section 3.3.1. This similar performance comes from the similar simulation environment with the only slight difference being traffic load (500 FTP sources in Scenario 1 versus 300 FTP sources plus 200 HTTP sources in Scenario 2), which make no major difference in the performance of the relationship between ECN-AQM and AQM.

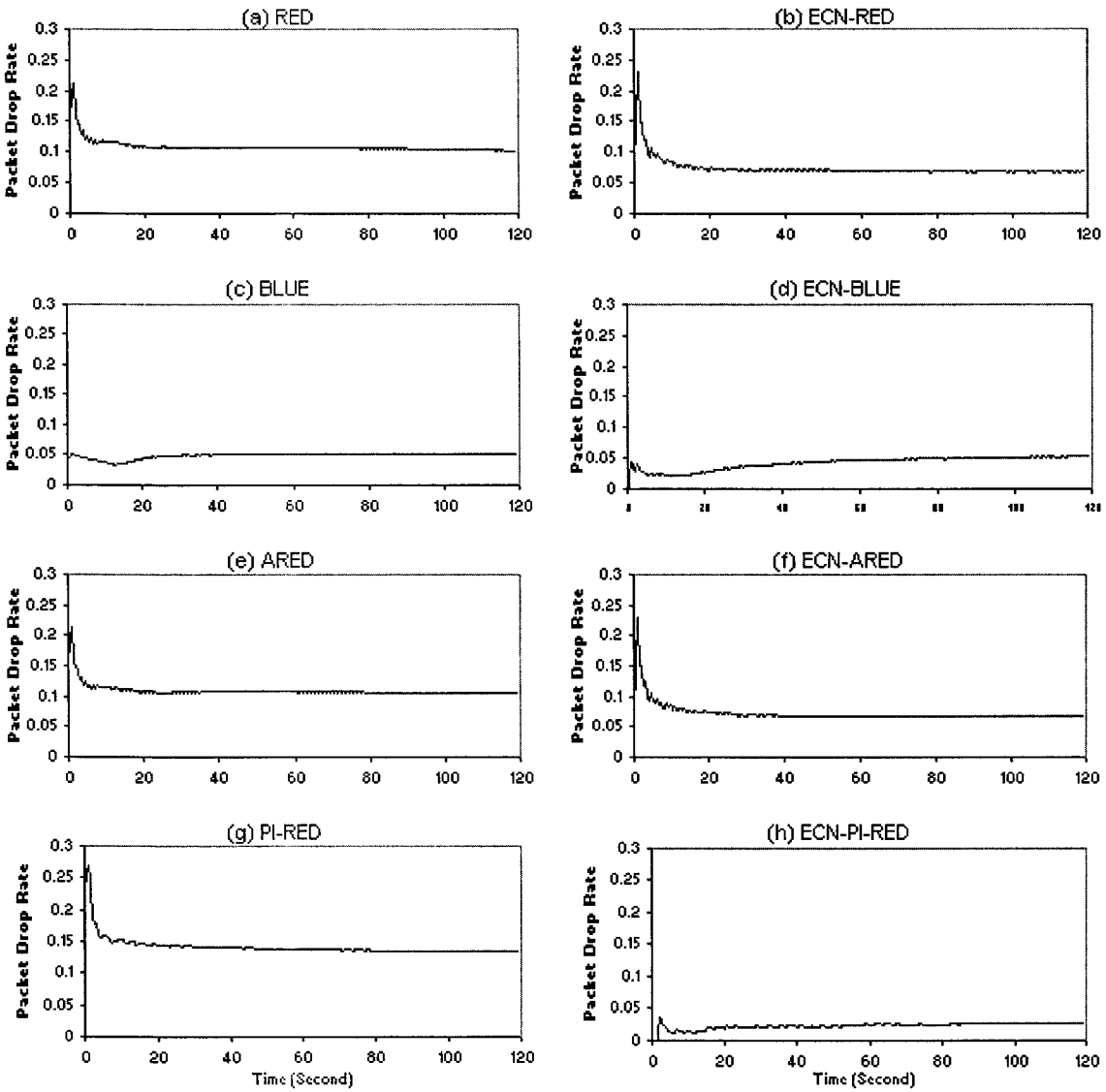

Figure 3-4 Average Packet Drop Rate Comparison of ECN-AQM with AQM for Scenario of 300-FTP-plus-200-HTTP in Single Bottleneck Network

The average packet drop rate comparisons of all those eight algorithms in Scenario 2 are shown in Figure 3-4. Likewise, even though the packet drop rates of AQM schemes in Scenario 2 are generally lower than in Scenario 1, the relationship between ECN-AQM and AQM is similar for both scenarios. That is, ECN-AQM, except for ECN-BLUE, generally shows lower packet drop rates than AQM. While ECN-BLUE does not exhibit any improvement on the packet drop rate when compared with BLUE, which has already kept its packet drop rate low with its own method. This may come from the fact that the different methods they use for reducing their packet drops make them unable to benefit from each other to further reduce their packet losses, when traffic load is relatively light.

3.3.3 Scenario 3 (100 FTP sources and 400 HTTP sources)

Figures 3-5 ~ 3-6 show the instantaneous queue size and the average drop rate comparisons of all of the algorithms in Scenario 3, respectively. We can see from Figure 3-5 that the instantaneous queue size performances of all of the investigated algorithms in Scenario 3 are similar to those in Scenario 2. However, the average packet drop rate performances of ECN-AQM schemes present differently. It shows that ECN can not help RED, BLUE and ARED improve their packet drop rates because ECN has little effect on this when the traffic load is much lighter in Scenario 3 (with only 100 FTP sources and 400 HTTP sources) than in Scenario 1 (with totally 500 FTP sources). However, ECN-PI-RED still exhibits a lower packet drop rate than PI-RED because its abilities to reduce packet drop rate is still strong even with light traffic load.

3.3.4 Average Packet Drop Rate Comparison Among Different Scenarios

To obtain a clear observation of the performance of all eight algorithms under the different simulation scenarios, we provide the average packet drop rate comparison among Scenarios1, 2 and 3 shown in Figure 3-7, as an example.

Figure 3-7 shows the relationship between the average packet drop rate and the different scenarios for all of the eight algorithms. It can be seen that the number of dropped packets in Scenario 3 is smaller than in Scenario 1 for all of the algorithms, which is due to the fact that most senders in Scenario 3 are short-lived sources (HTTP sources). Since they only send packets sporadically, the traffic load on the bottleneck router is lighter, resulting in

less discarded packets. Also, we have noticed that the packet drop rate of ECN-PI-RED is the lowest among the algorithms for the same reason as previously mentioned.

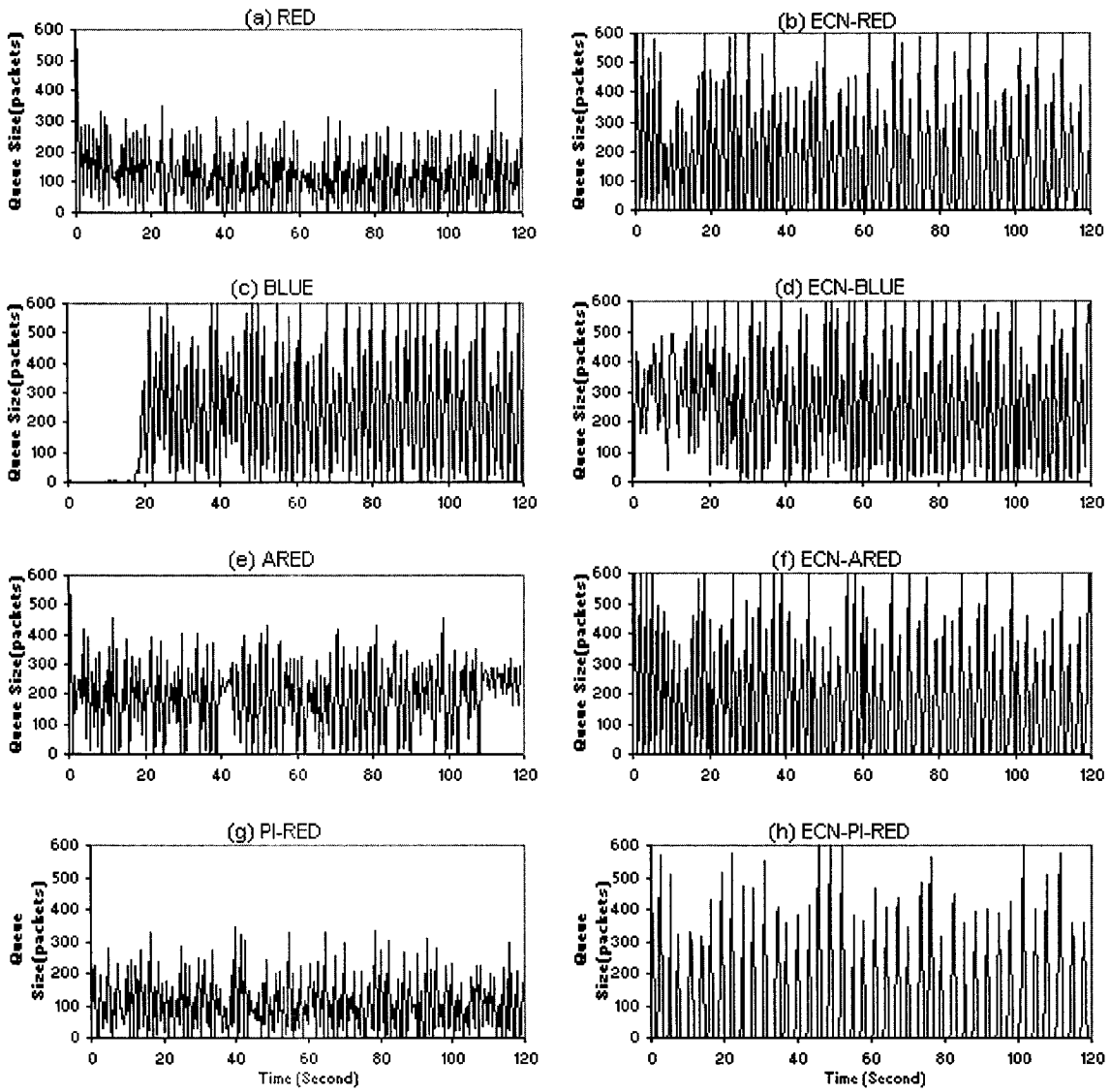

Figure 3-5 Instantaneous Queue Size Comparison of ECN-AQM with AQM for Scenario of 100-FTP-plus-400-HTTP in Single Bottleneck Network

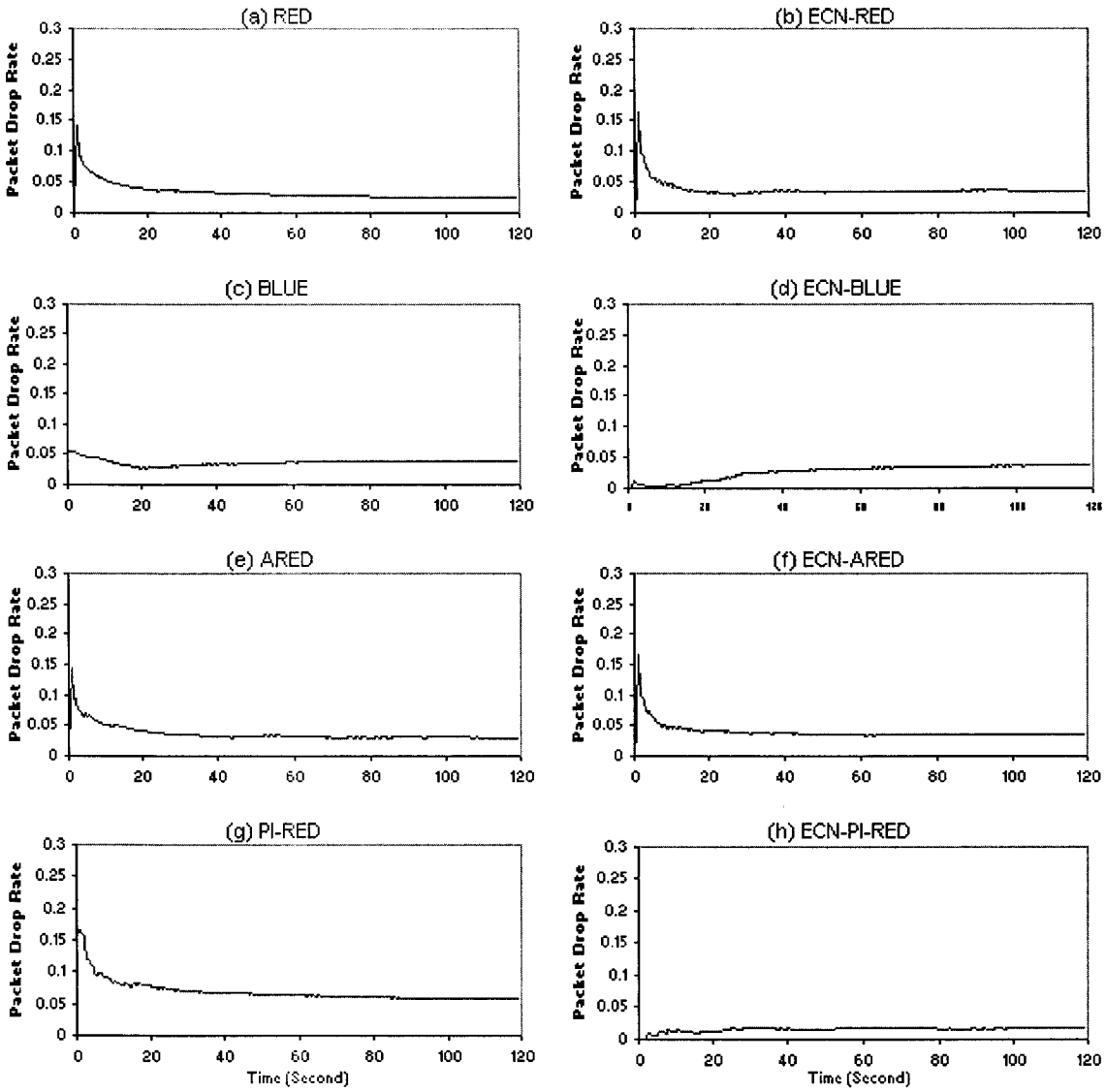

Figure 3-6 Average Packet Drop Rate Comparison of ECN-AQM with AQM for Scenario of 100-FTP-plus-400-HTTP in Single Bottleneck Network

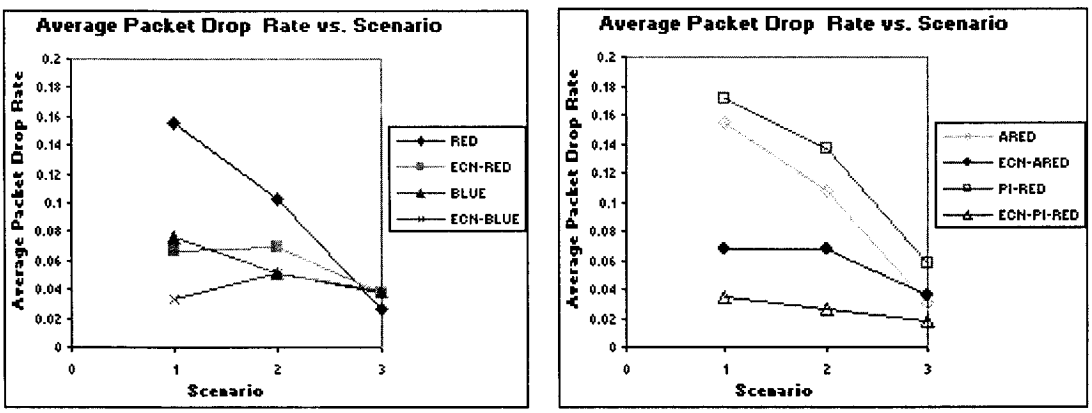

Figure 3-7 Average Packet Drop Rate versus Different Scenario for Different Algorithms

3.4 Concluding Remarks

Generally, ECN-AQM algorithms can greatly reduce unnecessary packet loss, but cannot stabilize their queue size. The exception is ECN-BLUE, which exhibits the similar performances to BLUE on queue size stability, as well as average packet drop rate in most cases.

Both ECN-RED and ECN-ARED perform similarly on queue size stability and average packet drop rate. So do RED and ARED. Obviously, ARED and ECN-ARED can not outperform RED and ECN-RED, respectively.

ECN-PI-RED can decrease the packet drop rate significantly, when compared with PI-RED and RED. However, its queue size oscillates severely, unlike PI-RED, which has a steady queue size. Therefore, the combination of ECN and PI-RED cannot solve both of the problems experienced by RED.

In addition to the factor of the algorithm itself, traffic load is another important factor, which affects the performances of the algorithms.

Chapter 4 ECN with AQM in Multi-bottleneck Network

We have investigated ECN with some AQM algorithms in the single bottleneck network model in the last chapter. This chapter will continue to study and compare the performance of these algorithms in the multi-bottleneck network model.

4.1 Simulation Studies

We use the multi-bottleneck network model, as shown in Figure 2-11 and described in Section 2.4.2, to do simulations and performance evaluation. We choose the simulated time of 120 seconds for each simulation run.

TCP-Reno is still implemented in the sources for the multi-bottleneck network. We use the same parameters for the file length probability distribution function of Equation (2-1) and the think time probability distribution function of Equation (2-2) as those in the single bottleneck network. The parameters of ECN-RED and RED, ECN-BLUE and BLUE, ECN-ARED and ARED, and ECN-PI-RED and PI-RED used for the multi-bottleneck network are also the same as those used in the single bottleneck network.

Based on the multi-bottleneck network model, we create two simulation scenarios, each with different numbers of FTP sources and HTTP sources, as shown in Table 4-1, in order to see how different types of TCP sources affect the performance of the algorithms. In each scenario, we observe the performance of RED when the combinations of router data service rates change, in order to study the general characteristics of our multi-bottleneck network. The different combinations used in this chapter are listed in Table 4-2 with the detailed meanings. We assign the different data service rates to the different routers to see how these rates affect the performance of the algorithms in the router.

Table 4-1 Number of FTP and HTTP Sources in Different Simulation Scenarios for Multi-bottleneck Network

Scenario No.	Number of FTP sources	Number of HTTP sources	Network configuration
4	700	0	Multi-bottleneck Network
5	200	500	Multi-bottleneck Network

Table 4-1 shows the number of FTP and HTTP sources in the different simulation scenarios for the multi-bottleneck network. These two scenarios are numbered 4 and 5 to distinguish from Scenarios 1 to 3 in the single bottleneck network. Scenario 4 contains 700 FTP sources. There are 200 FTP sources and 500 HTTP sources in Scenario 5 with all of the FTP sources coming from Source subnets 6 and 7.

Table 4-2 Different Combinations of Router Data Service Rates

Abbreviations	Meanings
45-45-45-45	Router 1 data service rate: 45 M bps; Router 2 data service rate: 45 M bps; Router 3 data service rate: 45 M bps; Router 4 data service rate: 45 M bps.
45-50-55-40	Router 1 data service rate: 45 M bps; Router 2 data service rate: 50 M bps; Router 3 data service rate: 55 M bps; Router 4 data service rate: 40 M bps.
45-50-50-45	Router 1 data service rate: 45 M bps; Router 2 data service rate: 50 M bps; Router 3 data service rate: 50 M bps; Router 4 data service rate: 45 M bps.
45-55-55-45	Router 1 data service rate: 45 M bps; Router 2 data service rate: 55 M bps; Router 3 data service rate: 55 M bps; Router 4 data service rate: 45 M bps.
45-60-60-45	Router 1 data service rate: 45 M bps; Router 2 data service rate: 60 M bps; Router 3 data service rate: 60 M bps; Router 4 data service rate: 45 M bps.
45-50-55-60	Router 1 data service rate: 60 M bps; Router 2 data service rate: 50 M bps; Router 3 data service rate: 55 M bps; Router 4 data service rate: 45 M bps.
45-55-50-45	Router 1 data service rate: 45 M bps; Router 2 data service rate: 55 M bps; Router 3 data service rate: 50 M bps; Router 4 data service rate: 45 M bps.
60-45-45-60	Router 1 data service rate: 60 M bps; Router 2 data service rate: 45 M bps; Router 3 data service rate: 45 M bps; Router 4 data service rate: 60 M bps.

The performance measurements used in this chapter are instantaneous queue size, average queue size, and average packet drop rate. All of these measurements are taken from each of the four routers.

4.2 Performance Evaluation in Scenario 4 (700 FTP sources and no HTTP sources)

In this section, we will present the simulation results obtained from 45-45-45-45 Network and 45-55-55-45 Network in the scenario of 700-FTP. The average queue size and the average packet drop rate performances of RED with the different combinations of data service rates will then be provided.

4.2.1 45-45-45-45 Network

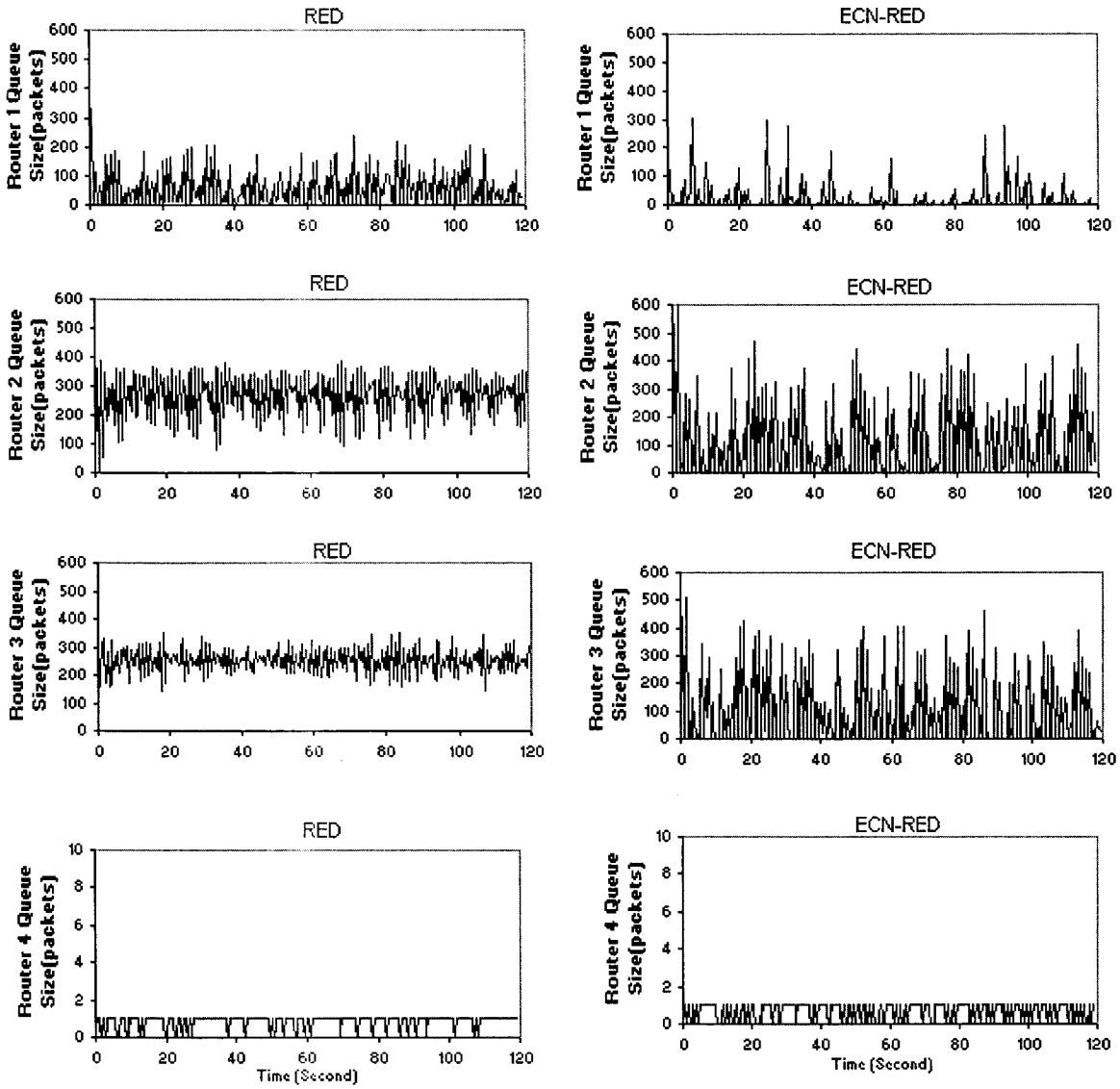

Figure 4-1 Instantaneous Queue Size Comparison of ECN-RED with RED with Router Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

Figure 4-1 shows the instantaneous queue sizes of RED and ECN-RED in all four routers, with the router data service rate of 45-45-45-45 in Scenario 4 (700 FTP sources). It can be seen that ECN-RED exhibits a little greater queue size fluctuation in Routers 1~3 than the corresponding RED algorithm, unlike the situation in the single bottleneck network, in which ECN-RED has much greater queue size fluctuation than RED. This difference comes from the different combinations of router data service rates, i.e., 45-45-45-45 in Scenario 4 versus 45-infinity-infinity-infinity in the case of the single bottleneck network, which is also a factor

that affects the performance of queue size in a certain router, in addition to the factor of traffic load. Observe also that the queue sizes of the algorithms in Router 4 are equally low (about 1 packet) mainly because most of the traffic in the network has been controlled (either dropped or blocked) in the upstream routers of Router 4, and therefore all incoming packets can be served in Router 4 instantly.

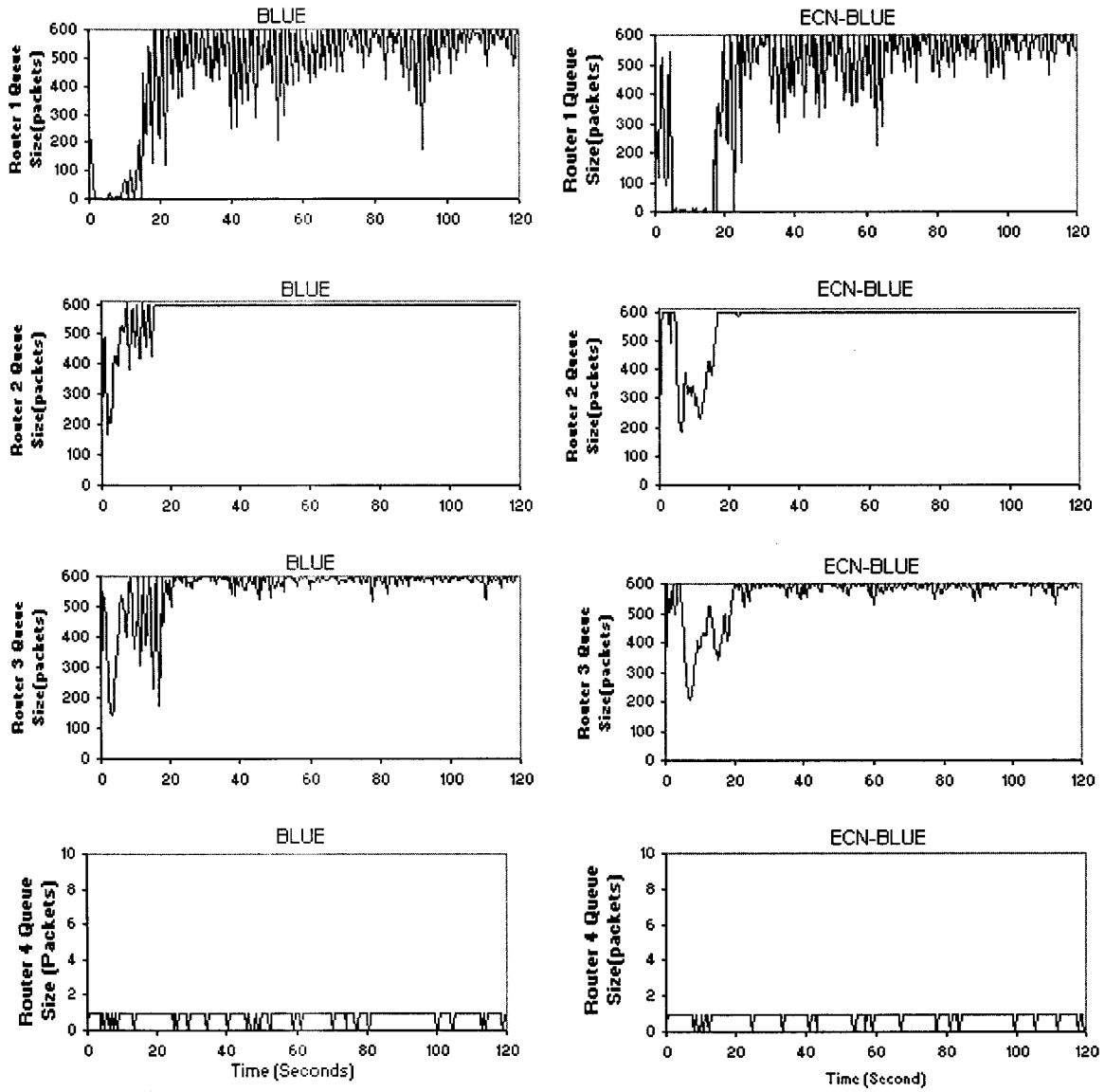

Figure 4-2 Instantaneous Queue Size Comparison of ECN-BLUE with BLUE with Router Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

Figure 4-2 shows the instantaneous queue sizes of BLUE and ECN-BLUE in all four routers, with the router data service rate of 45-45-45-45 in Scenario 4. As can be seen that

ECN-BLUE shows almost the same queue size fluctuation as BLUE, as in the single bottleneck network, because BLUE does not perform well in queue size stability according to its algorithm itself. Similarly, the queue sizes in Router 4 are about 1 packet for the same reason as mentioned before.

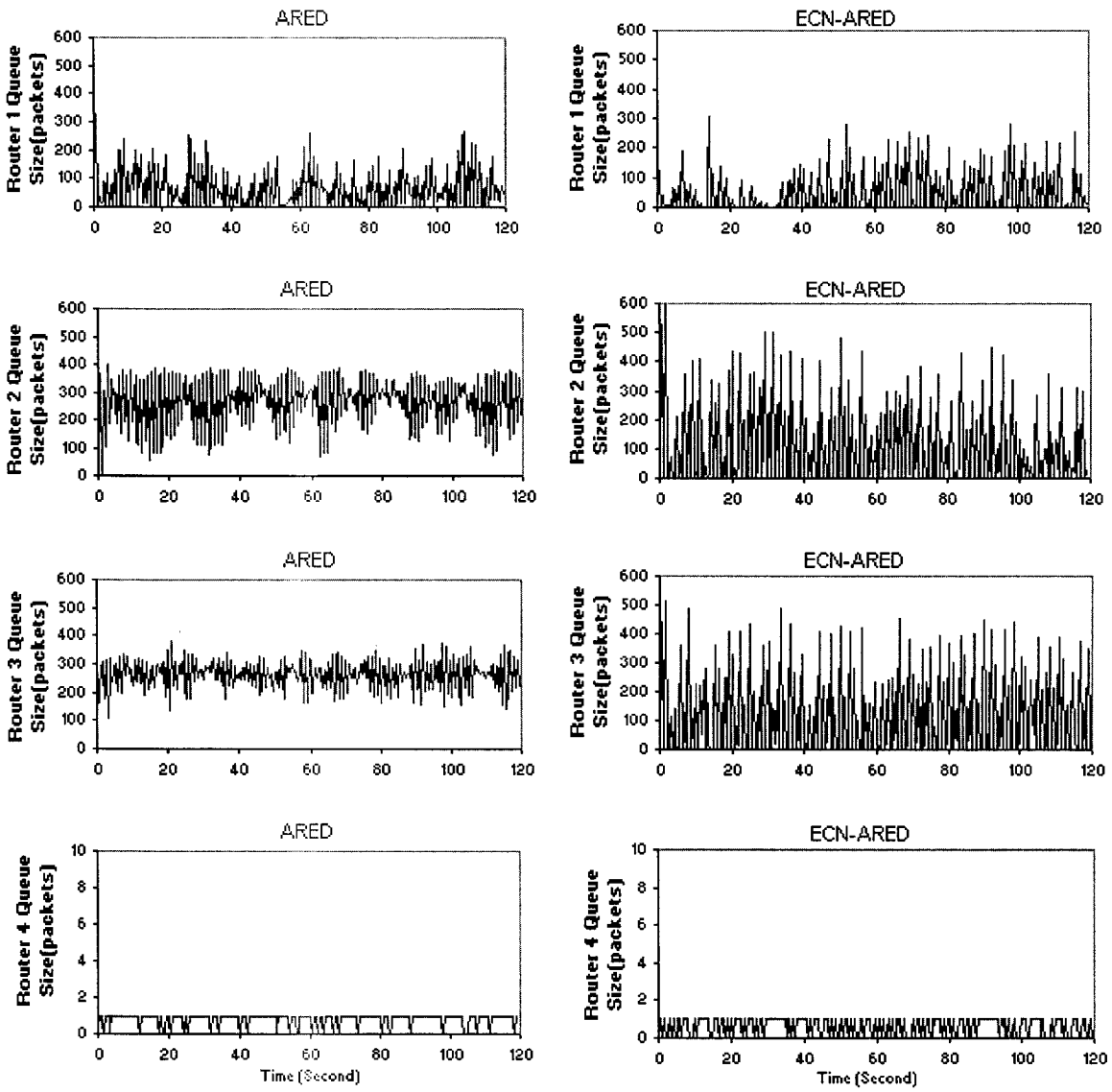

Figure 4-3 Instantaneous Queue Size Comparison of ECN-ARED with ARED with Router Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

Figures 4-3 shows the instantaneous queue sizes of ARED and ECN-ARED in all four routers, with the router data service rate of 45-45-45-45 in Scenario 4. Like ECN-RED, ECN-ARED exhibits a little wider queue size fluctuation in Routers 1~3 than the corresponding

ARED algorithm, which is unlike the situation in the single bottleneck network for the same reason as previously mentioned. Still, the queue sizes in Router 4 are as low as about 1 packet.

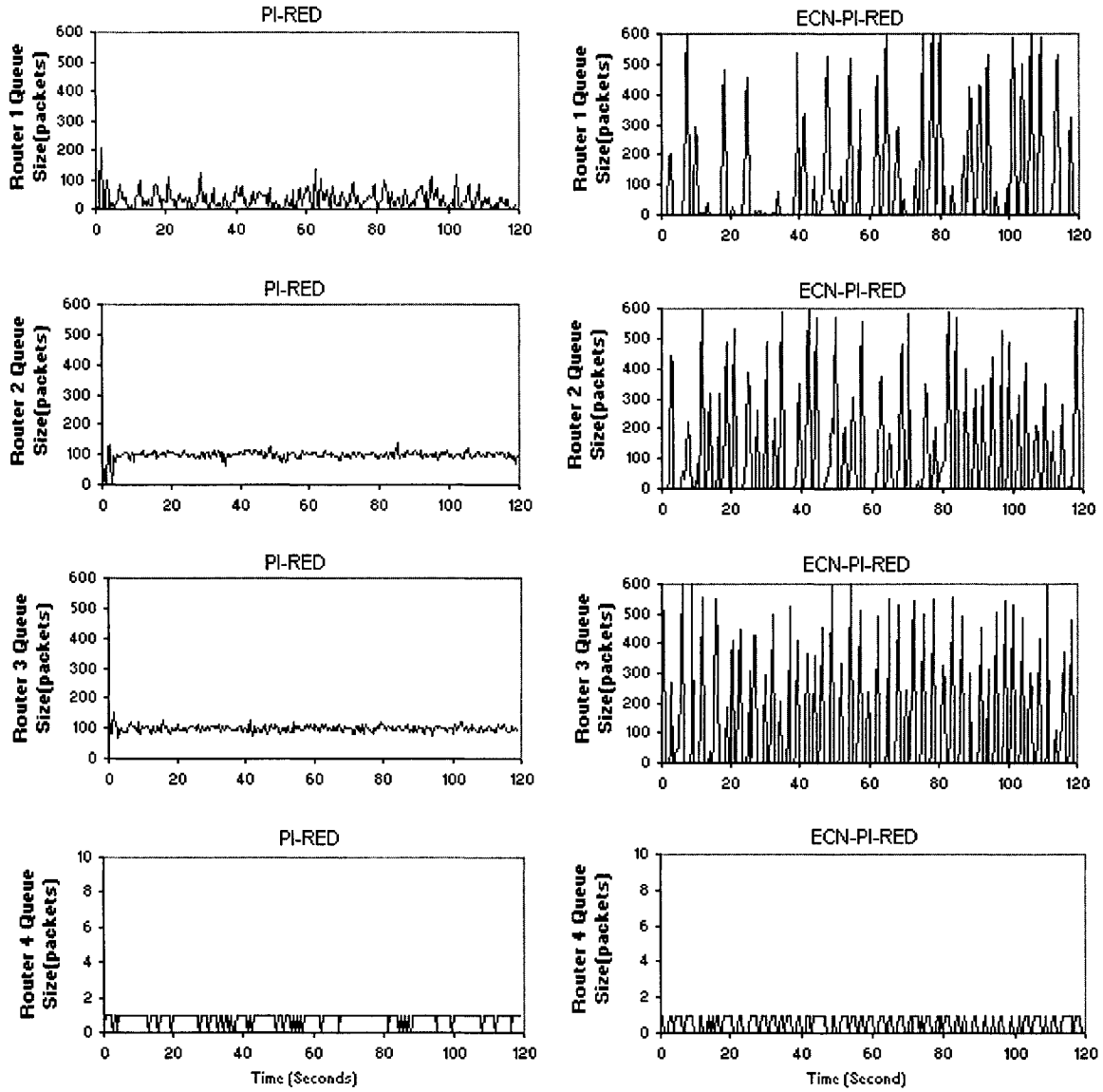

Figure 4-4 Instantaneous Queue Size Comparison of ECN-PI-RED with PI-RED with Router Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

Figures 4-4 shows the instantaneous queue sizes of PI-RED and ECN-PI-RED in all four routers, with the router data service rate of 45-45-45-45 in Scenario 4. From Figure 4-4, we can see that ECN-PI-RED presents extremely severe queue size fluctuation in Routers 1~3 when compared with PI-RED since PI-RED has a strong ability to control its queue size

at a reference level. Observe also that both of the algorithms have no queue built up in Router 4.

Table 4-3 Upper and Lower Bounds of 95% Confidence Interval of Average Queue Size (RED)

RED	Router 1 Confidence Interval	Router 2 Confidence Interval	Router 3 Confidence Interval	Router 4 Confidence Interval
Upper bound	63.13	271.42	259.15	0.88
Lower bound	57.33	267.42	255.17	0.86

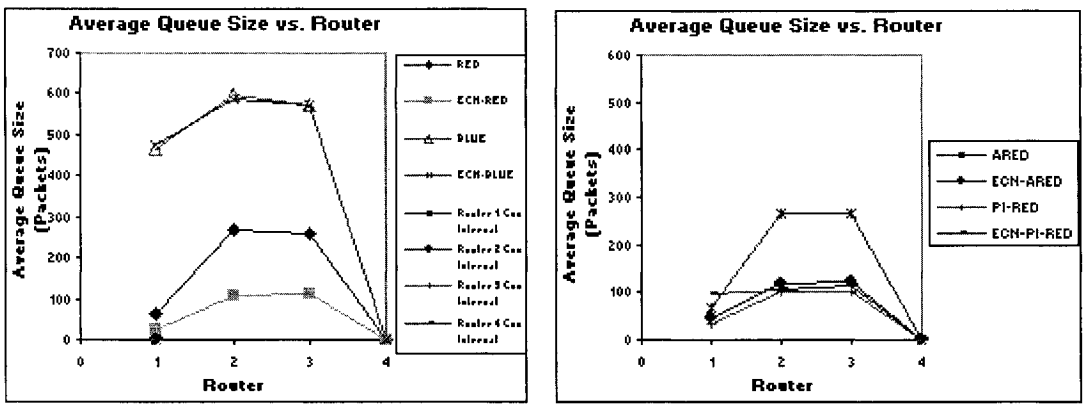

Figure 4-5 Average Queue Size versus Different Router for Different Algorithms with Router Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

For the performances of all of the algorithms on the average queue size and the average packet drop rate in all four routers with the router data service rate of 45-45-45-45 in Scenario of 700-FTP, we give Figures 4-5 and 4-6 to observe better by comparing them together. In the figures of depicting the relationship between the average queue size and the different router, and the relationship between the average packet drop rate and the different routers each individual point is the average of 4 different simulation runs. This enables us to obtain 95% confidence interval. In Figure 4-5, we show the 95% confidence interval of the average queue size of RED. The values of these upper and lower bounds of the interval are listed in Table 4-3. As can be seen that these bounds are very close. This is similarly observed in other simulations. For clarity purpose, we will just omit them for all of the other curves in this thesis.

Figure 4-5 shows the average queue size performances of all the eight algorithms with the router service rate combination of 45-45-45-45 in Scenario 4. Even though ECN-AQM schemes generally exhibit larger queue size fluctuation than AQM, as can be seen from Figures 4-1~ 4-4, thy present differently on their average queue sizes in Routers 1~3, due to the different algorithms: for ECN-RED and ECN-ARED, they have smaller average queue size than RED and ARED, respectively, while ECN-PI-RED builds up a little larger queues than PI-RED, but smaller queues than RED, and ECN-BLUE has almost the same queue size as BLUE. As for Router 4, all of the algorithms have no queue built up there for the same reason as mentioned before. In particular, we notice that all eight algorithms create the almost equally largest queues in Routers 2 and 3, where BLUE and ECN-BLUE build the large queues, RED and ARED build the midsize queues and the others build the small queues, according to the different algorithms. The relatively largest queues in Routers 2 and 3 come from the heavy traffic loads there with the FTP flows coming from six source subnets (totally 600 FTP sources, as shown Table 2-1). While in Router 1, its traffic load is lighter (500 FTP flows) than that in Routers 2 and 3, therefore the queues with a little smaller size are built there. Also, we have noticed that the average queue size performances of RED and ECN-RED are very similar to ARED and ECN-ARED, respectively.

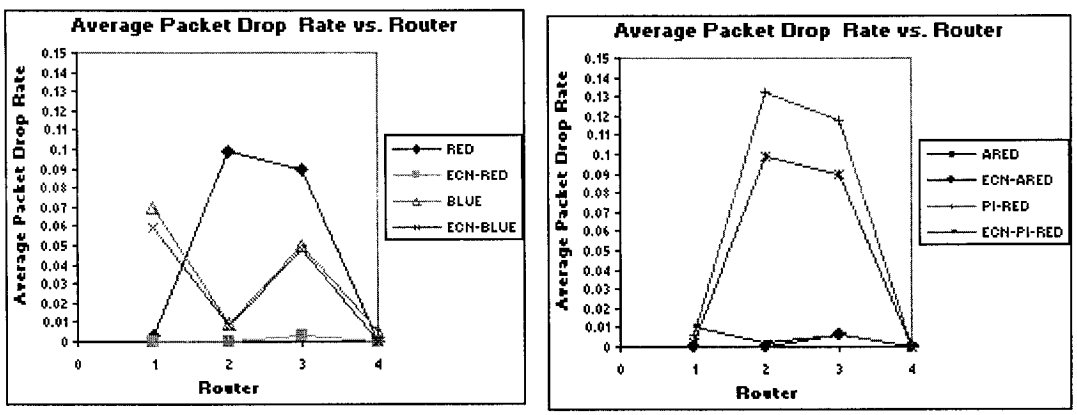

Figure 4-6 Average Packet Drop Rate versus Different Router for Different Algorithms with Router Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

Figure 4-6 shows the average packet drop rate performances of all eight algorithms with the combination of 45-45-45-45 in Scenario 4. As can be seen that ECN-RED, ECN-ARED and ECN-PI-RED are significantly superior to RED, ARED and PI-RED in

decreasing their packet drop rates, respectively, because they mark packets instead of dropping them. However, the application of ECN to BLUE cannot further reduce the packet drop. Also, we noticed an interesting pattern for both BLUE and ECN-BLUE: there is a relatively higher packet drop rate in Router 1, then a lower drop rate in Router 2, again a higher rate in Router 3, and no packet being discarded in Router 4. This phenomenon is in accordance with that presented in Figure 4-5, i.e., the packet drop rate for Router 1 is the highest, so its queue size is the smallest; Router 2 has the lowest packet drop rate, so its queue size is the largest; and both of the queue size and the packet drop rate for Router 3 are in the middle. Still, ARED has almost the same performance as RED, and the same thing happens to ECN-ARED and ECN-RED.

4.2.2 45-55-55-45 Network

We have observed the relatively high congestion level in Routers 2 and 3 from the 45-45-45-45 Network. In order to relieve this high congestion level, we increase the data service rates of these two routers to 55 Mbps for all those algorithms to see how they perform when the situation of the routers changes.

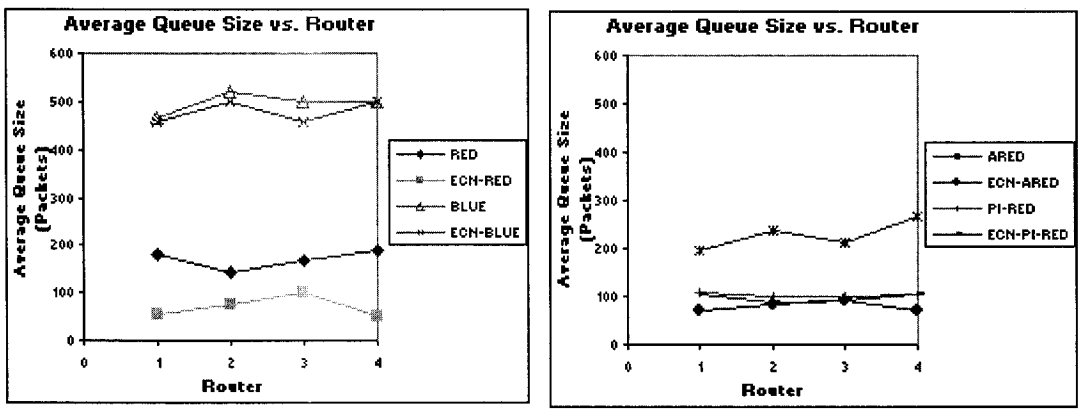

Figure 4-7 Average Queue Size versus Different Router for Different Algorithms with Router Data Service Rate of 45-55-55-45 in Scenario of 700-FTP

Unlike the situation of 45-45-45-45, when with the combination of 45-55-55-45, as can be seen from Figure 4-7, all ECN-AQM schemes exhibit smaller queue size than AQM schemes, and all eight algorithms experience the different levels of congestion in all of the four routers. Both unlikeness results from the factors such as the router data service rate, the traffic load, and the algorithm itself. It also shows that moderating the heavy congestion in

some routers like Routers 2 and 3 in this case by increasing their router data service rates will inevitably increase the congestion levels in their neighbor routers like Router 4. In addition, we still cannot see ARED and ECN-ARED respectively improve much than RED and ECN-RED.

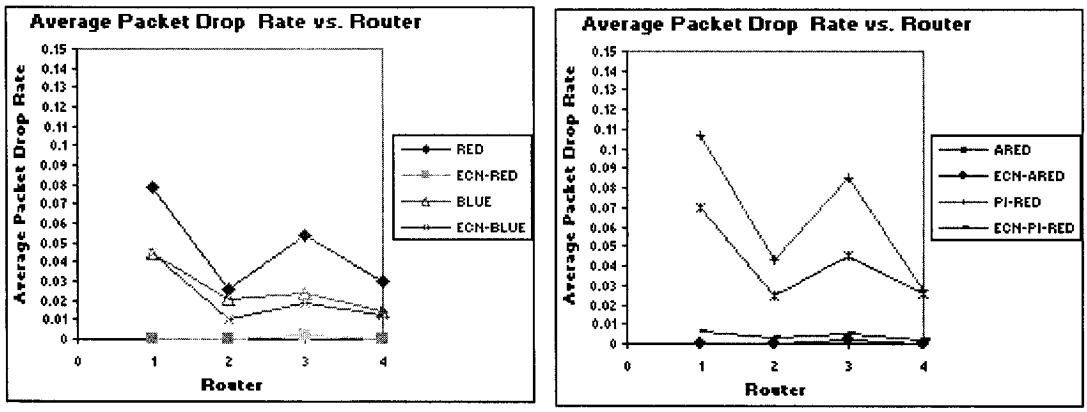

Figure 4-8 Average Packet Drop Rate versus Different Router for Different Algorithms with Router Data Service Rate of 45-55-55-45 in Scenario of 700-FTP

Figure 4-8 shows the average packet drop rate performances of all of the eight algorithms with the router service rate of 45-55-55-45 in Scenario 4. Not surprisingly, all of the ECN-AQM algorithms except for ECN-BLUE significantly outperform the original AQM algorithms on the packet drop rate performance, due to the packet marking strategy. More interesting, each curve in Figure 4-8 exhibits a similar pattern: a higher packet drop rate is in Router 1, then a lower drop rate is in Router 2, next a higher rate is in Router 3 than that in Router 2, and again a lower rate is in Router 4. This can probably be explained by the fact that more packets discarded in a router may generally result in less packets discarded in its downstream router since the traffic load may have been changed there. However, the similar phenomenon does not happen when with the combination of 45-45-45-45 as shown in Figure 4-6 because there are other factors such as the router data service rate, which may affect the performance of the algorithms.

4.2.3 RED Performance with Different Router Data Service Rates

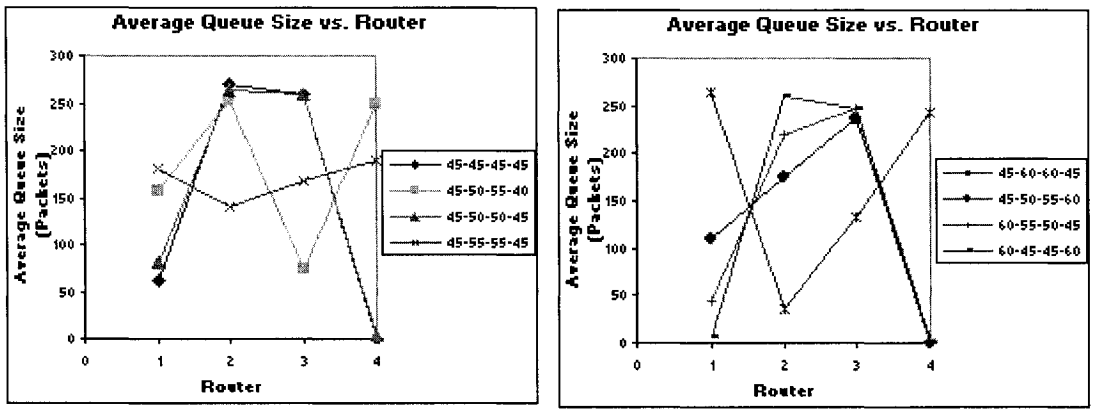

Figure 4-9 Average Queue Size versus Different Router for RED with Different Combinations of Router Data Service Rates in Scenario of 700-FTP

The RED algorithm presents differently on the average queue size when with the different combinations of router data service rates in Scenario 4 (700 FTP sources), as shown in Figure 4-9. We can observe four different situations. Like in Figure 4-7, RED experiences congestions in all of the four routers when with the combination of 45-55-55-45. When the combination is 45-50-55-40, even in Router 4 there is a queue built up, which is mainly because the service rate in Router 4 is too low to process all incoming packets. Compared with the other combinations, only under the combination of 45-60-60-45, the low level of congestion in Routers 2 and 3 is observed. The reason is simple: the service rates in these two routers are higher than the others. When with the rest of combinations, RED presents the similar performances, i.e., the relatively large queues are built in Routers 2 and 3, the medium queue is in Router 1 and almost no queue is in Router 4, for the same reason as stated for Figure 4-5. Obviously, the average queue size of RED in a router is generally quite sensitive to the data service rate in that router, when given a fixed router traffic load

The figures depicting the average packet drop rate of RED with the different combinations of router data service rates in Scenario 4, as shown in Figure 4-10, are respectively similar to those shown in Figure 4-9. This may come from the reason that packet drop rate is proportional to queue size to some extent for the algorithm like RED. Hence, the average packet drop rate of RED is also shown to be sensitive to the router service rate when the traffic load is fixed.

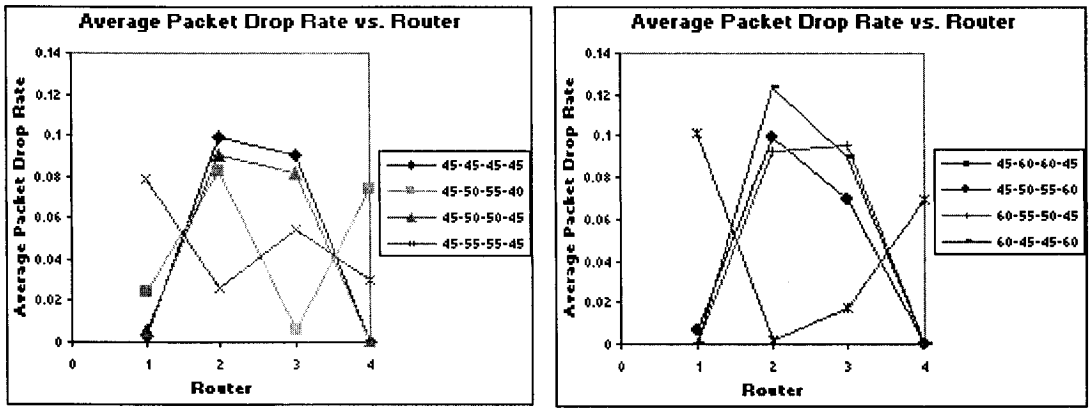

Figure 4-10 Average Packet Drop Rate versus Different Router for RED with Different Combinations of Router Data Service Rates in Scenario of 700-FTP

We also study the average queue size and the average packet drop rate performances of RED when the data service rate in one router changes while the service rates of the others remain the same rate.

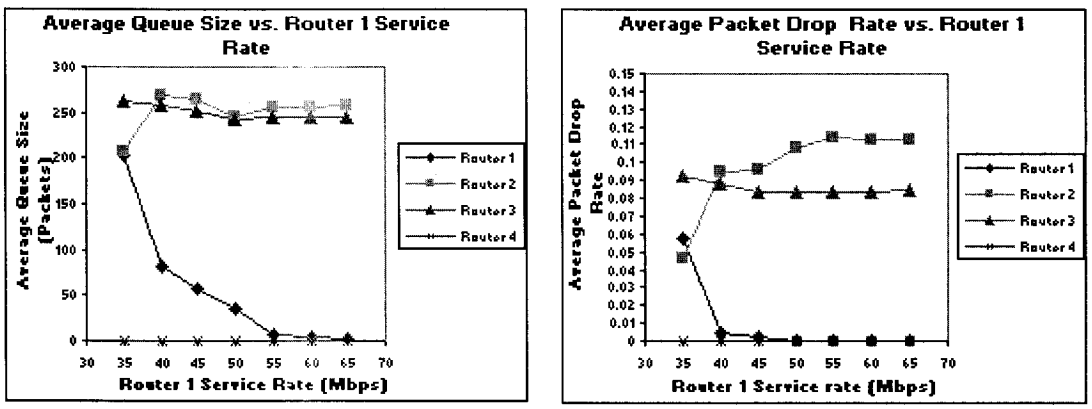

Figure 4-11 Average Queue Size and Average Packet Drop Rate of RED versus Different Service Rate in Router 1 with Same Rate of 45 M bps for Routers 2, 3, and 4 in Scenario of 700-FTP

Figure 4-11 show the average queue sizes and the average packet drop rates of RED in all of the four routers when the data service rate of Router 1 changes from 35 Mbps to 65 Mbps, and the service rates of Routers 2, 3 and 4 all remain the same rate of 45 Mbps (x-45-45-45, here "x" means changeable rate) in Scenario 4 (700 FTP sources). We can see that these two figures are similar. It is natural that both of the queue size and the packet drop rate of Router 1 decrease as its service rate increases since most packets are served without

waiting too long due to the increasing data service rate, and therefore not too many packets are dropped. Such performances in Router 1 then affect the performances of its downstream neighbor, Router 2. That is, both of the queue size and the packet drop rate of Router 2 increases as the service rate of Router 1 increases. This may come from the fact that, on the one hand, the whole system's capacity is enhanced by the increased router data service rate in Router 1, which encourages the source to send more packets into the network, which increases the traffic load, on the other hand, the decreasingly dropped packets in Router 1 cannot relieve the traffic load of its downstream router, therefore the downstream router presents the increasing queue size and packet drop if its service rate is not enhanced. Also, we notice that the queue sizes of Router 4 are all about 1 packet for the same reason as previously mentioned.

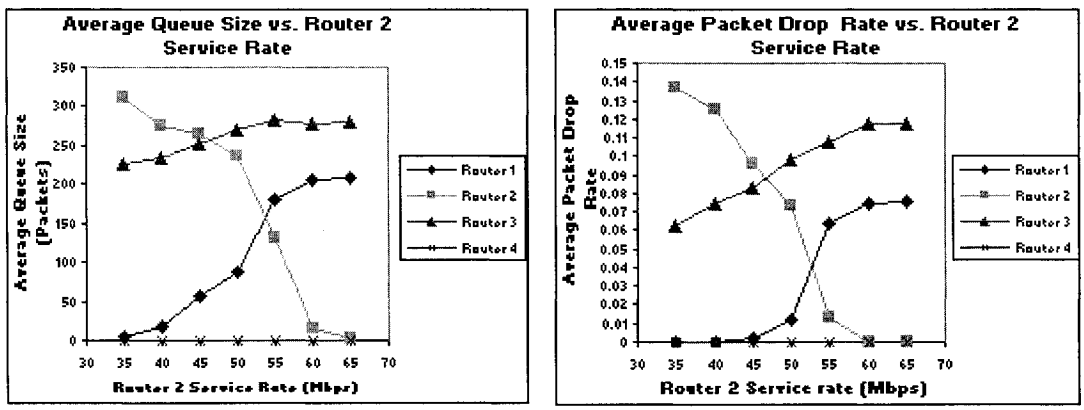

Figure 4-12 Average Queue Size and Average Packet Drop Rate of RED versus Different Service Rate in Router 2 with Same Rate of 45 M bps for Routers 1, 3, and 4 in Scenario of 700-FTP

Figure 4-12 show the average queue sizes and the average packet drop rates of RED in all of the four routers when the data service rate of Router 2 changes from 35 Mbps to 65 Mbps, and the service rates of Routers 1, 3 and 4 all remain the same rate of 45 Mbps (45-x-45-45) in Scenario 4. Likewise, it can be seen that both of the queue size and the packet drop rate of Router 2 decreases as its service rate increases, and the queue size and the packet drop rate of its downstream neighbor, Router 3, increases as the service rate of Router 2 increases, for the same reason as mentioned before. In addition, we observe that the queue size and the packet drop rate of Router 2's upstream neighbor, Router1, also increase as the service rate of Router 2 increases. This may be due to the fact that the increasing data processing ability in a

router increasingly enhances the processing ability of the whole system, which make the source increase its transmission rate to send more packets into the network, resulting in the increasing queue size in Router1, and more and more packets being dropped there. Still, the queue sizes of Router 4 are all about 1 packet.

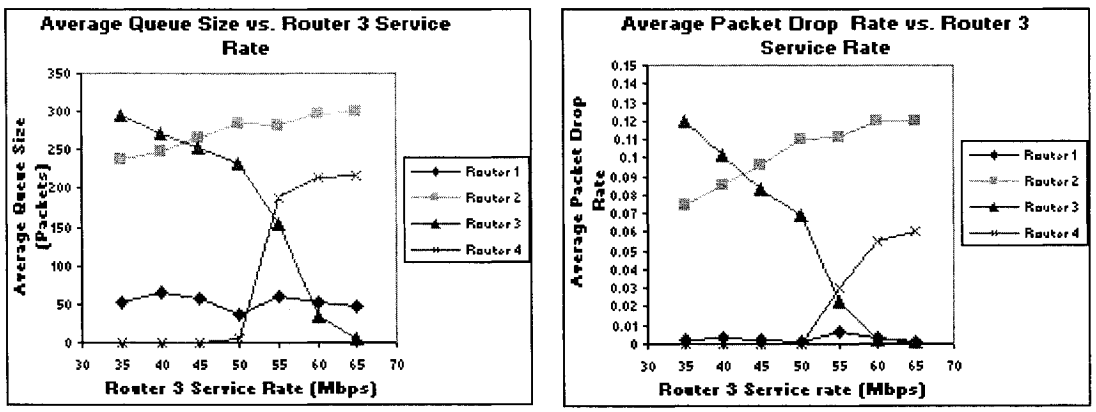

Figure 4-13 Average Queue Size and Average Packet Drop Rate of RED versus Different Service Rate in Router 3 with Same Rate of 45 M bps for Routers 1, 2, and 4 in Scenario of 700-FTP

Figure 4-13 show the average queue sizes and the average packet drop rates of RED in all of the four routers when the data service rate of Router 3 changes from 35 Mbps to 65 Mbps, and the service rates of Routers 1, 2 and 4 all remain the same rate of 45 Mbps (45-45-x-45) in Scenario 4. Similar performances on the relationship between the average queue sizes of all routers and the increasing data service rate of Router 3, and the relationship between the average packet drop rates of all routers and the increasing data service rate of Router 3 to those shown in Figures 4-12 are obtained, respectively.

Figure 4-14 show the average queue sizes and the average packet drop rates of RED in all of the four routers when the data service rate of Router 4 changes from 35 Mbps to 65 Mbps, and the service rates of Routers 1, 2 and 3 all remain the same rate of 45 Mbps (45-45-45-x) in Scenario 4. Similarly, both of the queue size and the packet drop rate of Router 4 decreases as its service rate increases, and the queue size and the packet drop rate of its upstream neighbor, Router 3, increases and then becomes stable as the service rate of Router 4 increases, for the same reason as mentioned before.

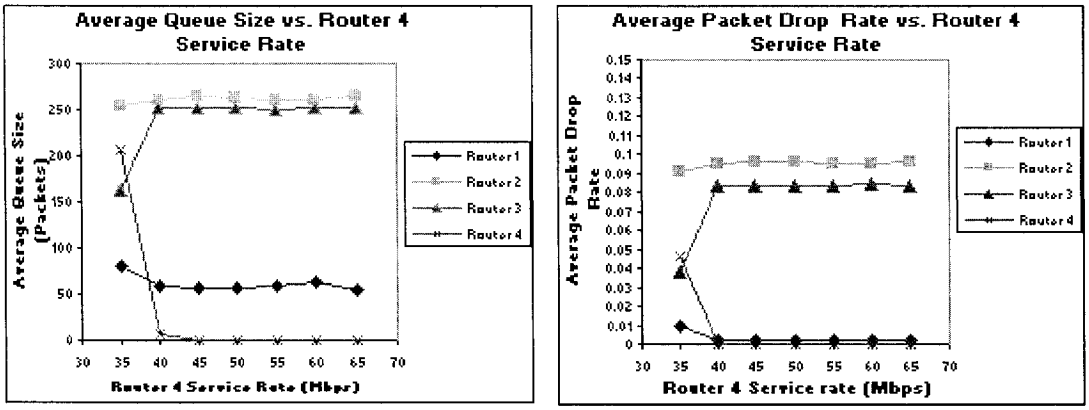

Figure 4-14 Average Queue Size and Average Packet Drop Rate of RED versus Different Service Rates in Router 4 with Same Rate of 45 M bps for Routers 1, 2, and 3 in Scenario of 700-FTP

4.3 Performance Evaluation in Scenario 5 (200 FTP source and 500 HTTP sources)

In this section, we will present the simulation results obtained from 45-45-45-45 Network in the scenario of 200-FTP-plus-500-HTTP. The average queue size and the average packet drop rate performances of RED with the different combinations of router data service rates will then be provided.

4.3.1 45-45-45-45 Network

The instantaneous queue size comparison of ECN-RED with RED in 45-45-45-45 Network for Scenario 5 (200 FTP sources and 500 HTTP sources) is shown in Figure 4-15. As can be seen, except that there is no queue built in Router 4, ECN-RED displays similarly great queue size fluctuation to RED in Routers 1~3, unlike the situation in Scenario 4. This is because the traffic load, which is lighter in Scenario 5 than in Scenario 4, is also a factor that affects the performance of the AQM algorithms, i.e., the more sporadic traffic make the queue sizes of AQM algorithm more unstable.

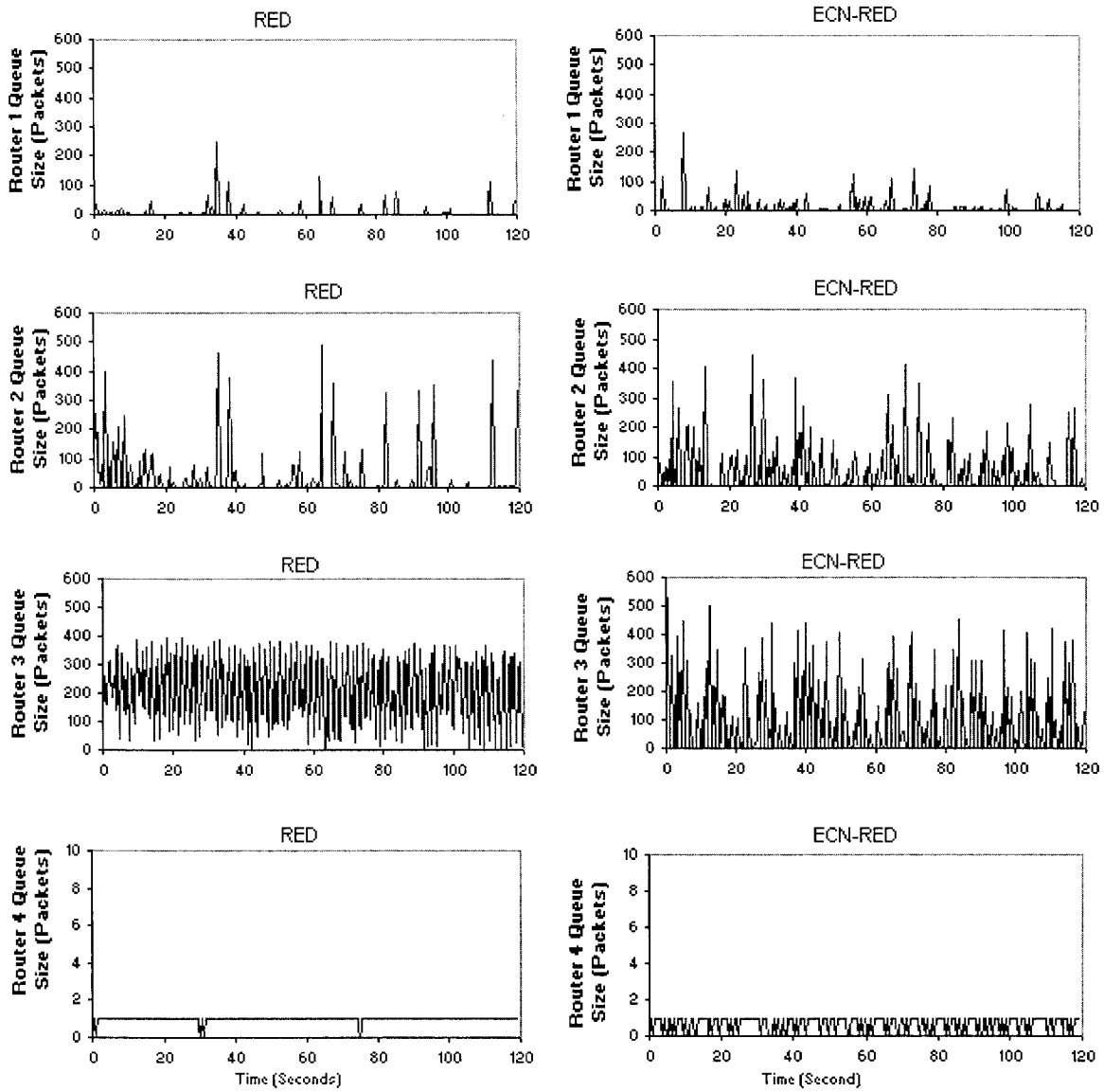

Figure 4-15 Instantaneous Queue Size Comparison of ECN-RED with RED with Router Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

The instantaneous queue size comparison of ECN-BLUE with BLUE in 45-45-45-45 Network for Scenario 5 is shown in Figure 4-16. Likewise, ECN-BLUE exhibits almost the same great queue size fluctuation as BLUE in Routers 1~3 for the same reason as previously mentioned.

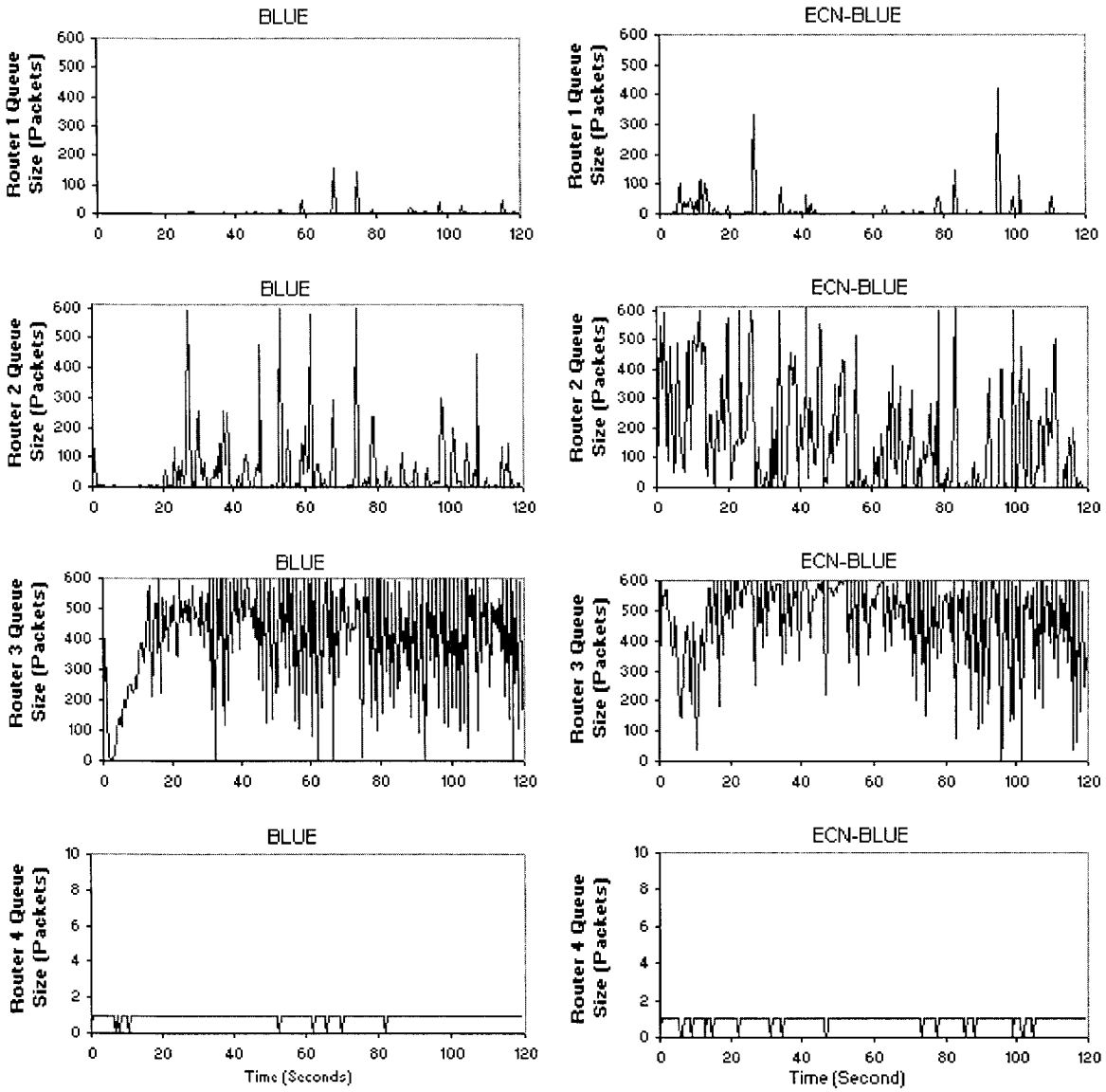

Figure 4-16 Instantaneous Queue Size Comparison of ECN-BLUE with BLUE with Router Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

Figure 4-17 shows the instantaneous queue size comparison of ECN-ARED with ARED in the 45-45-45-45 Network for Scenario 5. We can see that ECN-ARED performs similarly to ARED on the queue size fluctuation in Routers 1~3.

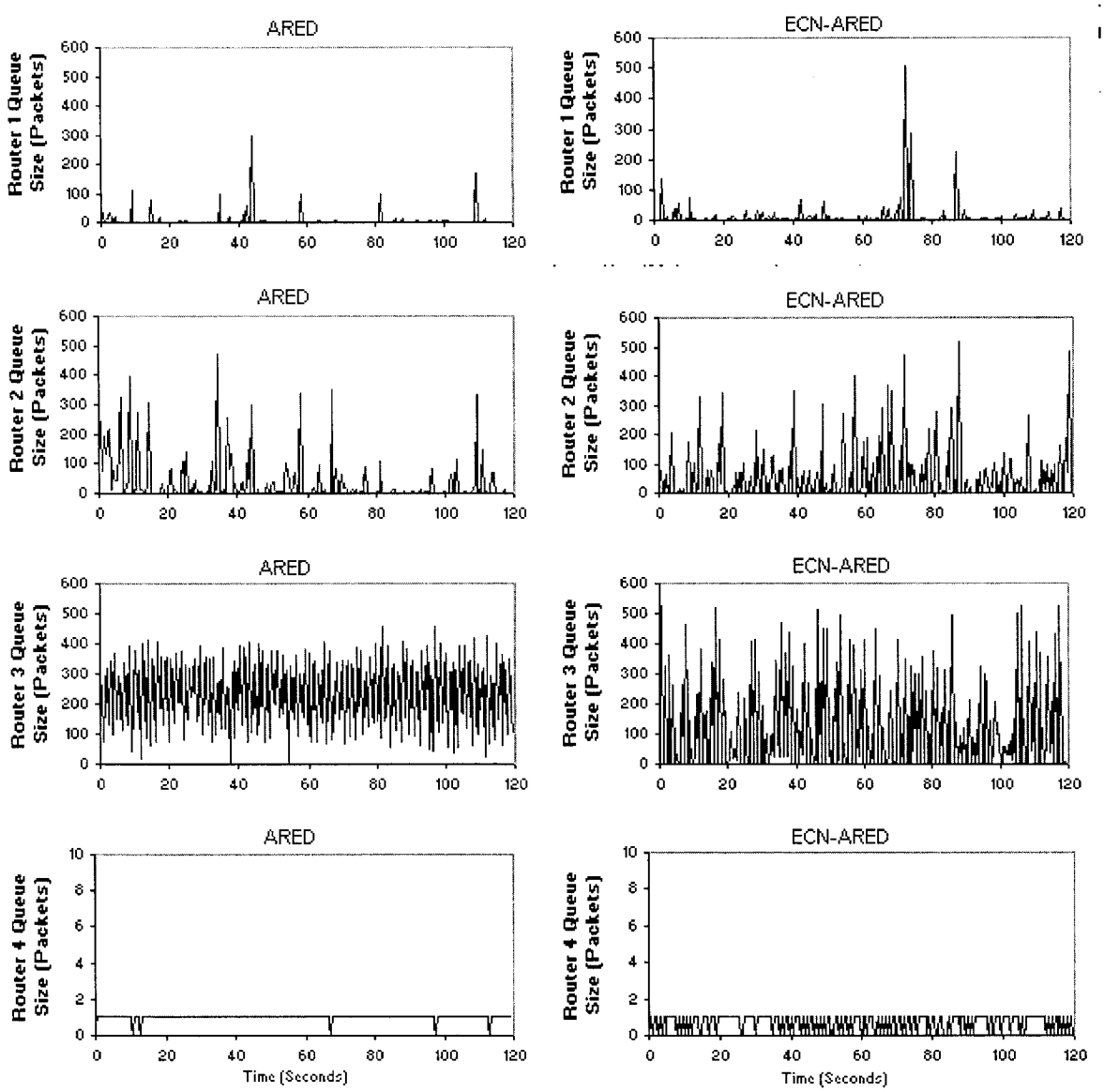

Figure 4-17 Instantaneous Queue Size Comparison of ECN-ARED with ARED with Router Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

Figure 4-18 shows the instantaneous queue size comparison of ECN-PI-RED with PI-RED in the 45-45-45-45 Network for Scenario 5. It can be seen that ECN-PI-RED cannot greatly improve the queue size fluctuation, unlike the situation in Scenario 4 for the same reason as mentioned before.

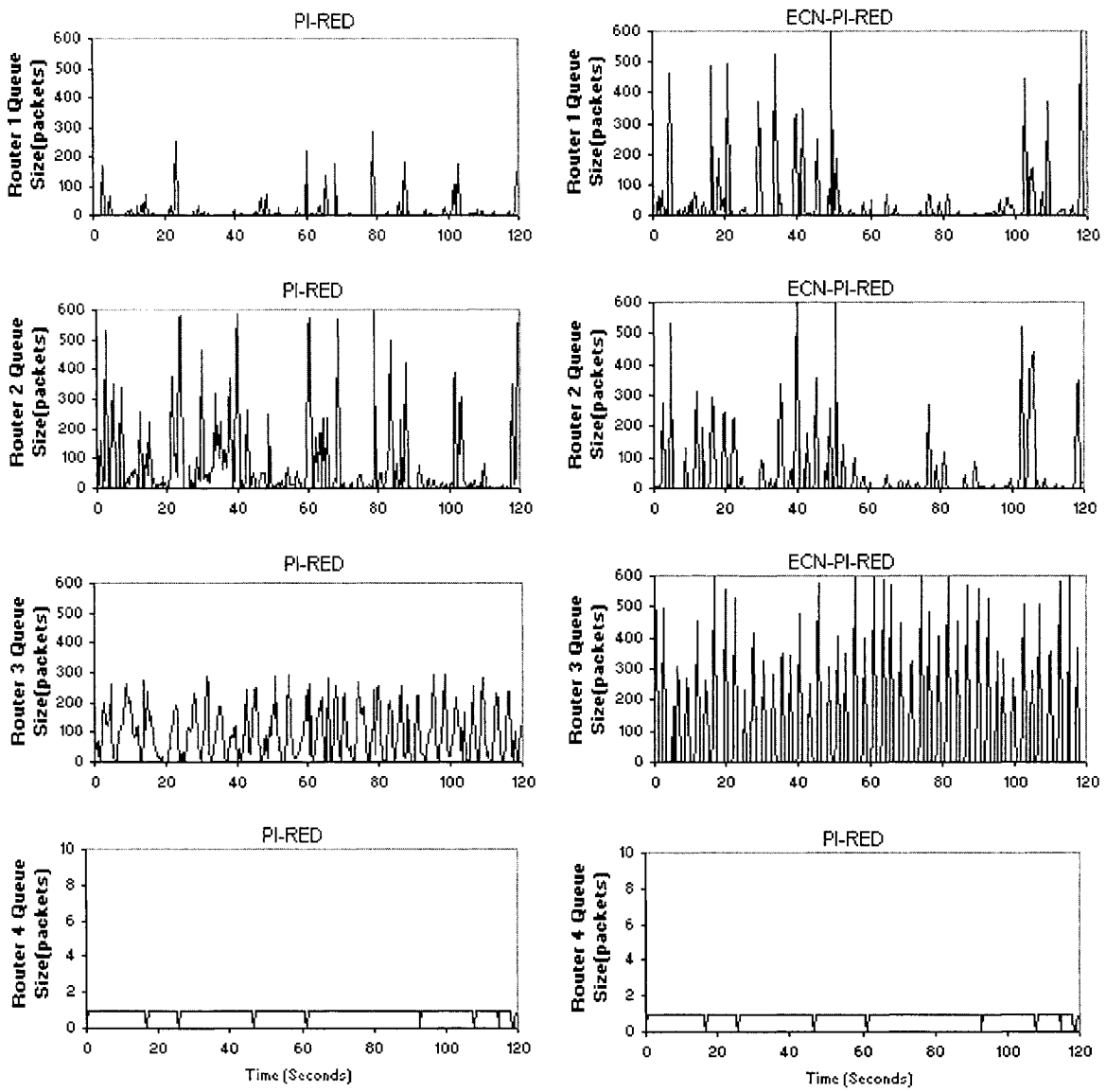

Figure 4-18 Instantaneous Queue Size Comparison of ECN-PI-RED with PI-RED with Router Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

Figure 4-19 shows the average queue sizes performance of all studied algorithms with the combination of 45-45-45-45 in Scenario 5. When compared with Figures 4-5 and 4-7, Figure 4-19 shows a different queue size performance: ECN-RED and ECN-ARED respectively presents lower queue size than ARED and RED only in Router 3 while their queue sizes in Router 2 are respectively larger than ARED and RED; For ECN-PI-RED, its average queue size in some routers can be greater than PI-RED, like the situation shown in Figure 4-5, or can be smaller than PI-RED, like the situation in Figure 4-7; However, ECN-

BLUE exhibits larger queue size than BLUE. The difference between Figure 4-19 and Figure 4-5 comes from the different traffic loads (there are 700 FTP sources for Figure 4-5 and 200 FTP plus 500 HTTP sources for Figure 4-19). While the difference between Figure 4-19 and Figure 4-7 comes from the combined factors of the different combinations of router date service rates (the combination of 45-55-55-45 in Figure 4-7 versus the combination of 45-45-45-45 in Figure 4-19) and the different traffic loads. Observe also that all algorithms build up the relatively small queues in Routers 1 and 2 and the largest queues only in Router 3, which is slightly different from the situation in Scenario 4 as shown in Figure 4-5, in which Routers 2 and 3 build up equally large queues due to the same heavy loads (600 FTP flows) there. This difference results from the fact that, in Scenario 4, all long-lived flows (200 FTP flows coming from Source subnets 6 and 7) pass through Router 3 with the other short-lived flows (400 HTTP flows coming from Source subnets 1~4), while only half of long-lived flows (100 FTP flows coming from Source subnet 6) plus the same amount of the short-lived flows as that passing through Router 3 passing through Router 2. By comparing each of the curves in Figure 4-19 with that in Figure 4-5, we observe that all of the algorithms shown in Figure 4-19 generally create smaller queues in Routers 1~ 3 than those shown in Figure 4-5, which is because the total traffic load in Scenario 5 (Figure 4-19) is lighter than that in Scenario 4 (Figure 4-5).

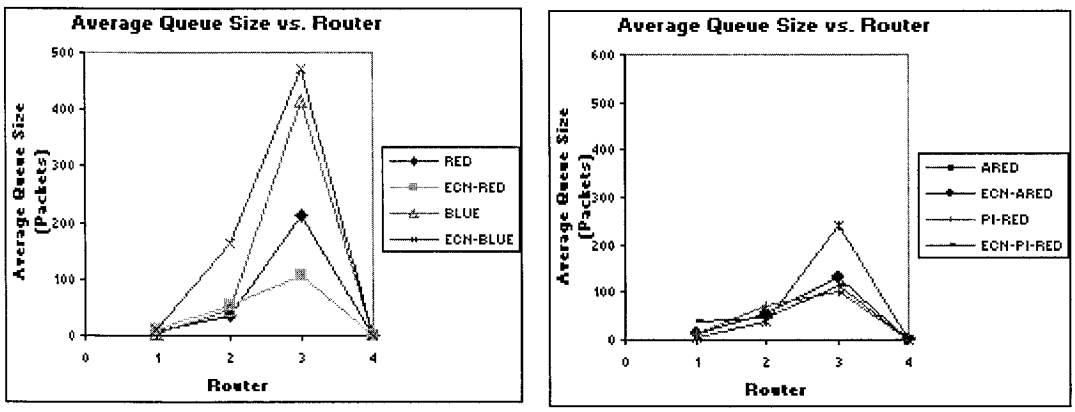

Figure 4-19 Average Queue Size versus Different Router for Different Algorithms with Router Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

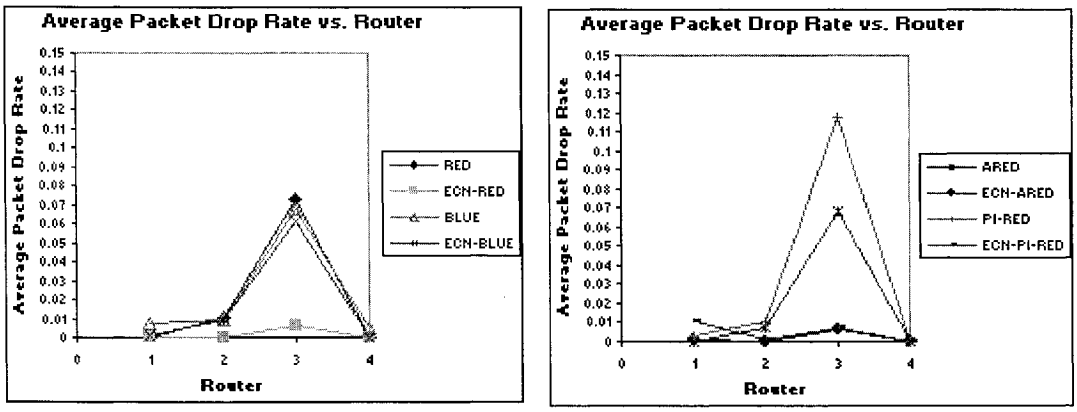

Figure 4-20 Average Packet Drop Rate versus Different Router for Different Algorithms with Router Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

Figure 4-20 shows the average packet drop rate performances of all of the eight algorithms with the router service rate of 45-55-55-45 in Scenario 5. Still, ECN-AQM algorithms except for ECN-BLUE present lower packet drop rates than the corresponding AQM algorithms, especially in Router 3, because of the ECN mechanism. Especially, ECN-PI-RED still presents extremely lower packet drop rate than PI-RED, while in Router 1, it has a different case due to the fact that ECN-PI-RED has a larger queue size than PI-RED in Router 1. Besides, we observe that all of the curves in Figure 20 are similar to each other. Still, ARED does not outperform RED, and neither does ECN-ARED.

4.3.2 RED Performance with Different Router Data Service Rates

Three different situations on the average queue size performances of RED with the different combinations of router data service rates in Scenario 5 can be observed from Figure 4-21. First, with the combinations of 45-50-50-45 and 60-55-50-45, RED builds the largest queues in Router 4 mainly because the service rate there is the lowest. Second, with the combinations of 60-45-45-60 and 45-50-55-60, RED builds up the largest queues in Router 3 and no queue in Router 4. The main reason for this also lies on the service rates set in those routers. Third, when with the rest of combinations, RED creates the largest queue sizes in Router 4 because of its low data service rates. Likewise, the average queue size of RED in Scenario 5 is sensitive to the data service rate in that router when the router traffic load is fixed.

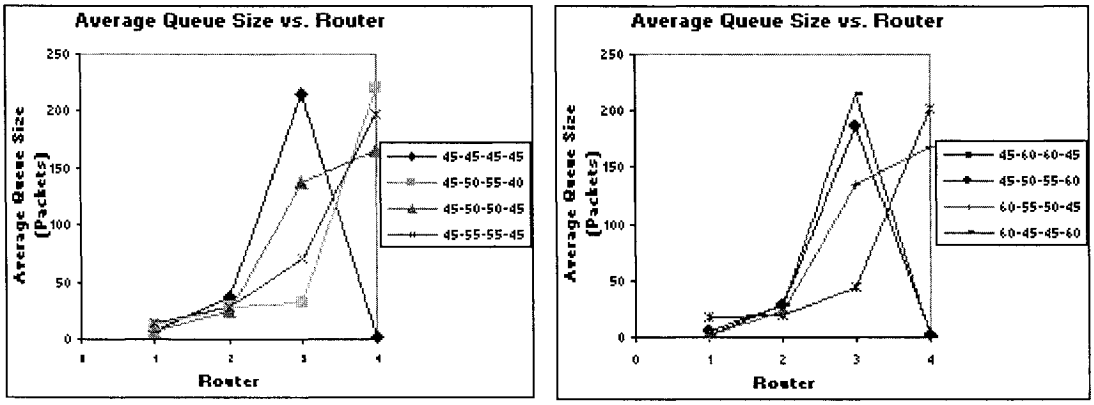

Figure 4-21 Average Queue Size versus Different Router for RED with Different Combinations of Router Data Service Rates in Scenario of 200-FTP-plus-500-HTTP

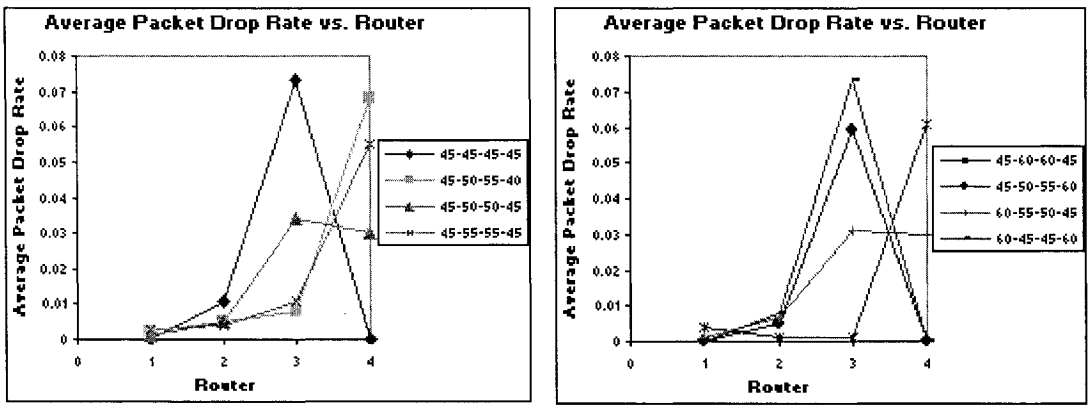

Figure 4-22 Average Packet Drop Rate versus Different Router for RED with Different Combination of Router Data Service Rates in Scenario of 200-FTP-plus-500-HTTP

As shown in Figure 4-22, the average packet drop rates of RED when with the different combinations of router data service rates in the scenario of 200-FTP-plus-500-HTTP are respectively similar to those in Figure 4-21, like the situation in Scenario 4. For almost each corresponding curve pair, the larger the router queue size, the higher the packet drop rate in that router. Still, the average packet drop rate of RED, like its queue size, is sensitive to the router service rate.

4.4 Concluding Remarks

The effect of ECN on our investigated AQM algorithms in the multi-bottleneck network is generally similar to that in the single bottleneck network. That is, ECN-AQM algorithms cannot stabilize its queue sizes, but generally present lower packet drop rates than AQM

schemes. The exception is ECN-BLUE, which not only cannot improve the performance of queue size stability, but also cannot much improve the performance of average packet drop rate, when compared with BLUE.

ARED can not get better performances than RED in terms of average queue size, average packer drop rate, and queue size stability, and the same observation can be obtained from the comparison of ECN-ARED with ECN-RED.

ECN-PI-RED is extremely superior to the original PI-RED and RED algorithms in the performance of packet drop rate. However, its queue size oscillates severely, unlike PI-RED, which presents a steady queue size. Therefore, the combination of ECN and PI-RED cannot solve both of the problems experienced by RED.

Our simulation experience regarding the performance of RED with different combinations of router data service rates indicates that, a RED router would have a relatively larger queue size and a higher packet drop rate (given a fixed traffic load in the router), as the router data service rate is lowered.

Chapter 5 DH-RED and DH-BLUE

In this chapter, we will apply the head dropping policy to reduce the delay of the conveyance of congestion information, as a solution to the problem of queue oscillation encountered by RED and its variants. Specifically, we will introduce two algorithms: DH-RED (Drop Head RED) algorithm and DH-BLUE (Drop Head BLUE) algorithm. Then the performance evaluation of these algorithms will be provided. Finally, the analysis of the tradeoff between DH-RED and ECN-RED, and the tradeoff between DH-BLUE and ECN-BLUE, will be presented.

5.1 The Drop Head Mechanism

The key idea of the drop head mechanism is to apply head dropping to lower the delay in conveying congestion information to the TCP source from the bottleneck router. In other words, the router starts to send the congestion signal to the source immediately after the onset of congestion is detected, instead of waiting for the time period of the total queuing delay. As a result, the source can react earlier to duplicate ACKs by invoking congestion control actions. This mechanism shortens the congestion period and reduces the number of sources that enter into the *Fast Retransmit* and *Fast Recovery* procedure. Consequently, it can stabilize the queue size, improve the throughput and decrease the packet loss at the same time. Obviously, the more percentage of queuing delay occupies in the total round trip delay, the more stable queue system will present in theory according to the above analysis. Furthermore, we expect DH-BLUE would outperform DH-RED on the performance of queue stability since BLUE well improves the unnecessary packet loss problem observed in RED. While our analysis is for RED and BLUE, we expect this head dropping strategy to remain effective for the other AQM schemes implemented in gateway routers.

The details of RED and BLUE will be provided in Appendixes A.1 and A.2. Here, we only present the detailed descriptions for DH-RED and DH-BLUE.

5.1.1 DH-RED

Our DH-RED algorithm uses a low-pass filter with an exponential weighted moving average to calculate average queue size avg. Like RED, this is then compared to two preset parameters, the minimum queue threshold $th_{\min}$ and the maximum threshold $th_{\max}$.

1) If $avg < th_{\min}$, the router always queues incoming packets at the tail of the queue and no packet has to be dropped.
2) If $avg > th_{\max}$, every incoming packet will be queued at the tail of the queue and the packet at the head of the queue will be removed.
3) If $th_{\min} \leq avg < th_{\max}$, an incoming packet will be inserted at the tail of the queue and the packet at the head of the queue will be removed with a probability (a function of the average queue size) computed by RED. However, if it is determined that the packet should not be dropped, it is inserted at the tail of the queue.

The above procedures function when the queue size is less than the buffer size. Otherwise, any arriving packets will be inserted at the tail of the queue and the packet at the head of the queue will be removed.

5.1.2 DH-BLUE

The general operation of the DH-BLUE algorithm is similar to that of BLUE. However, upon the arrival of a packet, DH-BLUE would queue the packet at the tail of the queue, and drop the packet from the head of the queue with a preset packet drop probability, which is the same probability used by BLUE for packet tail-dropping. The packet is queued at the tail of the queue if no packet drop decision is made for the packet.

Still, the above procedures function when the queue size is less than the buffer size. Otherwise, any arriving packets will be inserted at the tail of the queue and the packet at the head of the queue will be removed.

5.2 Simulation Studies

We use both of the single bottleneck network model shown in Figure 2-4, and the multi-bottleneck network model shown in Figure 2-11, to do our performance evaluation for both DH-RED and DH-BLUE algorithms.

All network systems are simulated for a time period of 120 seconds. TCP-Reno is implemented in the sources in our simulation model. Each TCP link is configured with the same propagation delay of 0.01 seconds.

The main simulation parameters of RED and DH-RED, and BLUE and DH-BLUE are the same as those of RED and ECN-RED, and BLUE and ECN-BLUE in our previous examination. That is, the minimum queue size threshold is 150 packets, the maximum queue threshold is 400 packets, the maximum packet drop rate is 0.1, the queue weight is 0.002, the initial drop probability is 0.05, the freeze time period is 0.01, the increase drop probability is 0.00025, and the decrease drop probability is 0.000025.

The performance measurements used in this chapter are instantaneous queue size and average packet drop rate, which have been defined in Section 3.3.

5.3 Performance Evaluation in Single Bottleneck Network

We create two scenarios based on the single bottleneck model: Scenario A, in which all TCP connections (500 connections in total) are FTP connections, and Scenario B, in which there are 300 FTP sources and 200 HTTP sources.

5.3.1 Scenario A (500 FTP sources)

Figure 5-1 shows the instantaneous queue sizes of RED, DH-RED, BLUE and DH-BLUE with 500 FTP sources in Scenario A. It can be seen from Figure 5-1 that the queue size of RED fluctuates with a large amplitude and the situation is even worse in BLUE. On the contrary, under the regulations of DH-RED and DH-BLUE, the queue sizes eventually become stable, with much smaller fluctuations when compared with RED and BLUE, due to the application of head dropping, especially for DH-BLUE, as predicted. It is obvious that the shortened RTT definitely makes the system more stable according to the analysis we did in the last section. However, DH-BLUE exhibits a relatively high initial value because no packet is dropped if the router buffer is not full and all of the sources send packets aggressively in the beginning of simulation.

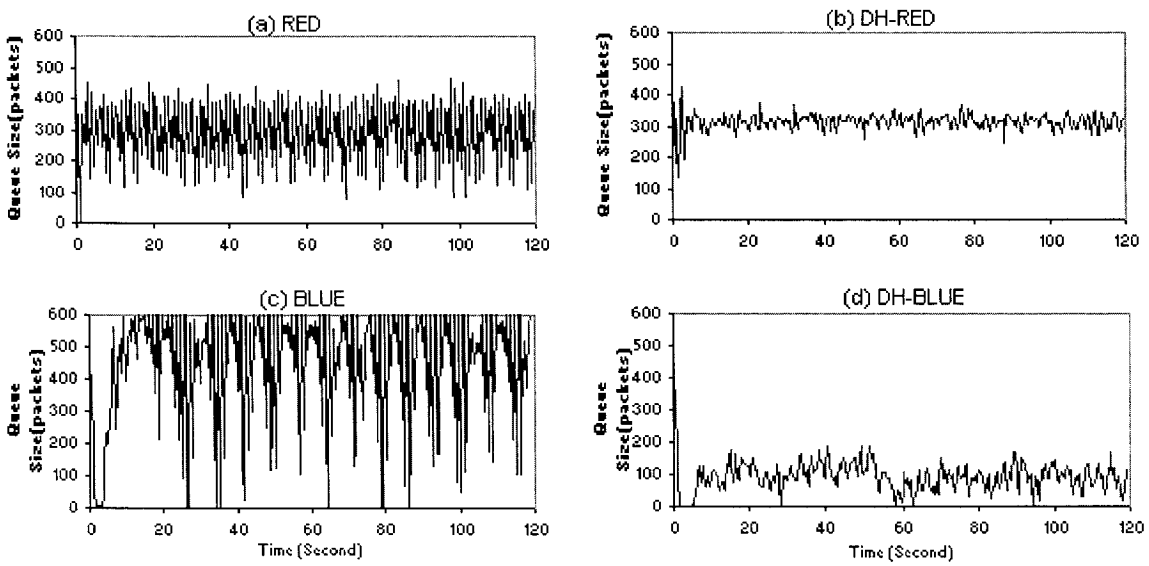

Figure 5-1 Instantaneous Queue Size Comparison of DH-AQM with AQM in Scenario of 500-FTP

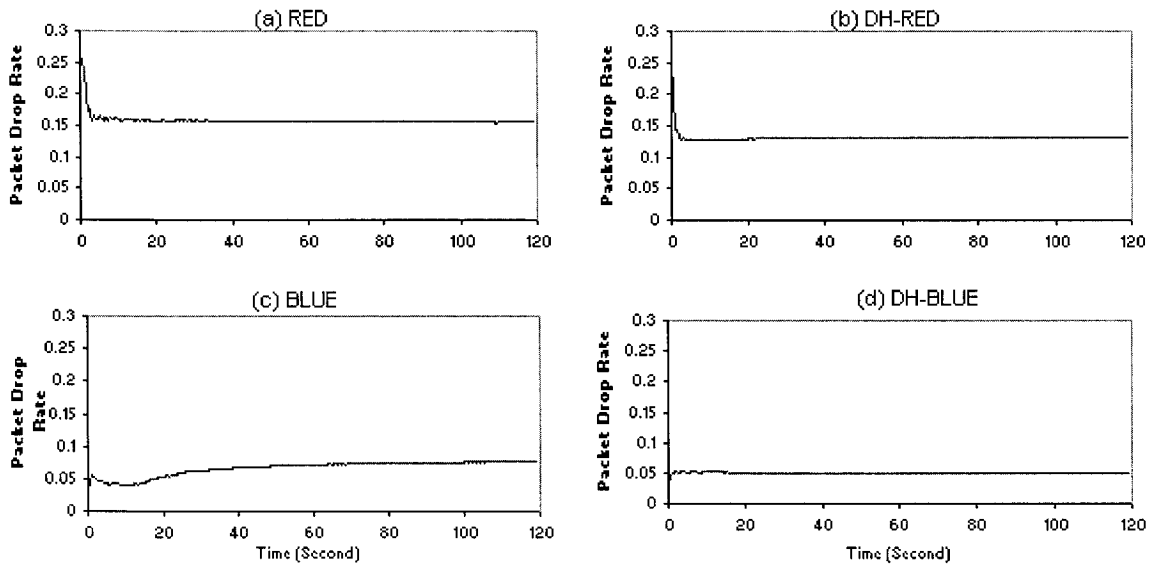

Figure 5-2 Average Packet Drop Rate Comparison of DH-AQM with AQM in Scenario of 500-FTP

Figure 5-2 shows the average packet drop rates of RED, DH-RED, BLUE and DH-BLUE with 500 FTP sources in Scenario A. It is not surprising that the packet loss of BLUE is much lower than that of RED. Also, we observe that both DH-RED and DH-BLUE have a lower packet drop rate on the average than RED and BLUE respectively because of the application of the head dropping policy.

Since the application of head dropping is aimed at stabilizing queue size, measuring queue size oscillation is of our particular interest in this chapter. We will give Figure 5-3 to observe how the queuing delay shortened in DH-RED and DH-BLUE affects their queue size stability performances.

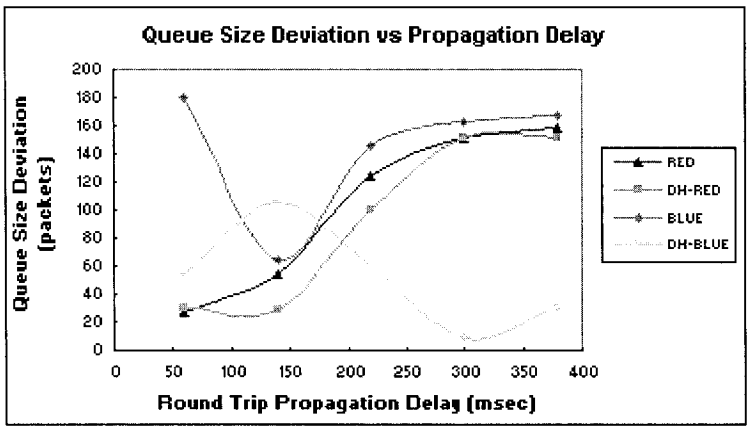

Figure 5-3 Queue Size Deviation versus Round Trip Propagation Delay in Scenario of 500-FTP

Figure 5-3 shows the relationship between queue size deviation and different round trip propagation delay in the scenario of 500-FTP. Observing that the queue size deviation of DH-RED is smaller than that of RED when the round trip propagation delay is less than 300 microseconds because DH-RED can reduce the delay of the transmission of congestion notification. They are similar to each other when the delay is longer than 300 microseconds. This is because the queuing delay has become unimportant in the queue size stability in the presence of large delays. Similarly, DH-BLUE shows a better performance than BLUE, i.e., the queue size deviation of DH-BLUE is much smaller than that of BLUE regardless how much the propagation delay is except when the range of round trip propagation delay is between 100 and 170 microseconds. The reason is that, in addition to the effect of the head dropping strategy on reducing the delay of the transmission of congestion notification, the improved packet loss in DH-BLUE is also a factor that reduces the total delay.

5.3.2 Scenario B (300 FTP sources and 200 HTTP sources)

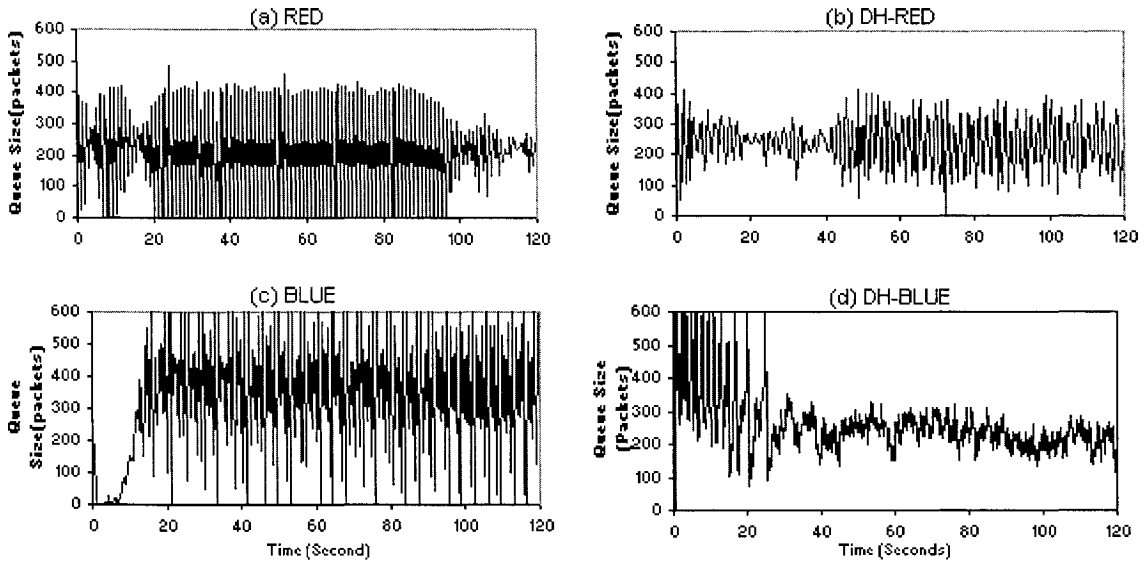

Figure 5-4 Instantaneous Queue Size Comparison of DH-AQM with AQM in Scenario of 300-FTP-plus-200-HTTP

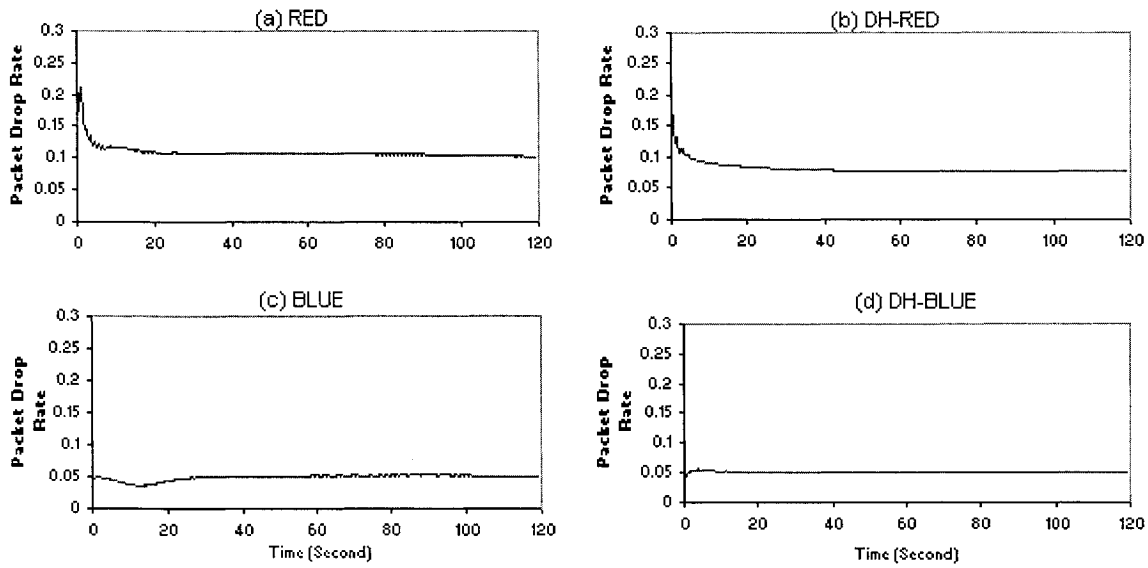

Figure 5-5 Average Packet Drop Rate Comparison of DH-AQM with AQM in Scenario of 300-FTP-plus-200-HTTP

Figure 5-4 shows the instantaneous queue sizes of RED, DH-RED, BLUE and DH-BLUE with 300 FTP sources and 200 HTTP sources in Scenario B. From Figure 5-4, we can see that RED shows obvious queue size oscillation, and the queue size of BLUE fluctuates even more widely, increasing to more than 600 packets and some buffer overflow occur. While

under the regulations of DH-RED and DH-BLUE, the queue sizes eventually become relatively stable when compared with RED and BLUE, especially for DH-BLUE (even it exhibits a little higher initial queue size for the same reason as previously mentioned).

Figure 5-5 shows the average packet drop rates of RED, DH-RED, BLUE and DH-BLUE in Scenario B. As can been see from Figure 5-5, DH-RED exhibit lower packet drop rate than RED. However, DH-BLUE shows similar packet drop rate to BLUE.

We have observed that the number of the dropped packets is smaller in Scenario B than in Scenario A (Figure 5-2 versus Figure 5-5), which is due to the fact that some TCP senders in Scenario B are short-lived sources (HTTP sources). Since they only send packets sporadically, the load to the bottleneck router is lighter, resulting in less discarded packets.

5.4 Performance Evaluation in Multi-bottleneck Network

We create two scenarios based on the multi-bottleneck network. There are totally 700 FTP source in Scenario C, and 200 FTP sources and 500 HTTP sources in Scenario D. Recall that the tradeoff between DH-AQM and ECN-AQM is also of our interest. Therefore, in this section, in addition to presenting the comparison of DH-AQM with AQM, we will present these with ECN-AQM. Specifically, we will compare RED, DH-RED with ECN-RED, and compare BLUE, DH-BLUE with ECN-BLUE.

5.4.1 Scenario C (700 FTP sources and no HTTP source)

Figure 5-6 shows the instantaneous queue sizes of RED, DH-RED and ECN-RED with the data service rate of 45-45-45-45 in Scenario C (700 FTP sources). It can be seen from Figure 5-6 that DH-RED presents a relatively more stable queue size than RED in Routers 1~3, due to the application of head dropping. It is obvious that the shortened RTT definitely makes the system more stable according to the analysis we did before. However, ECN-RED exhibits much larger queue size fluctuation than both of RED and DH-RED. Observe also that the queue sizes of all of the algorithms in Router 4 are equally low (about 1 packet) mainly because most traffic in the network has been controlled (either dropped or blocked) in the upstream routers of Router 4, and therefore all incoming packets can be served in Router 4 instantly.

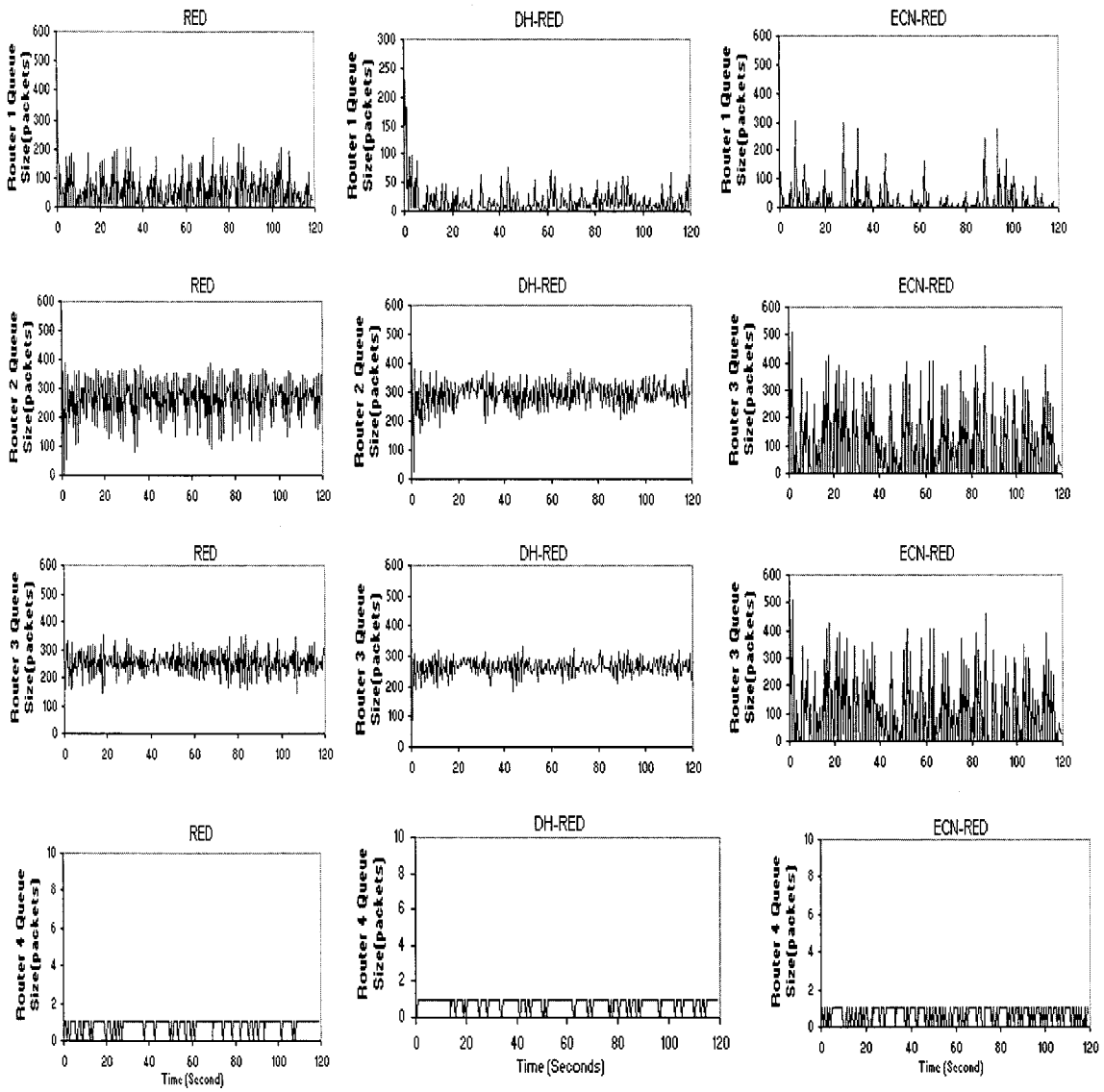

Figure 5-6 Instantaneous Queue Size Comparison of RED, DH-RED with ECN-RED with the data service rate of 45-45-45-45 in Scenario of 700-FTP

Figure 5-7 shows the instantaneous queue sizes of BLUE, DH-BLUE and ECN-BLUE, with the data service rate of 45-45-45-45 in Scenario C. We can see that even though DH-BLUE exhibits a relatively high initial value for the same reason as mentioned before, it exhibits a much stable queue size when compared with BLUE, due to the strategy of head dropping and the already well-reduced unnecessary packet loss. However, ECN-BLUE oscillates as much as BLUE.

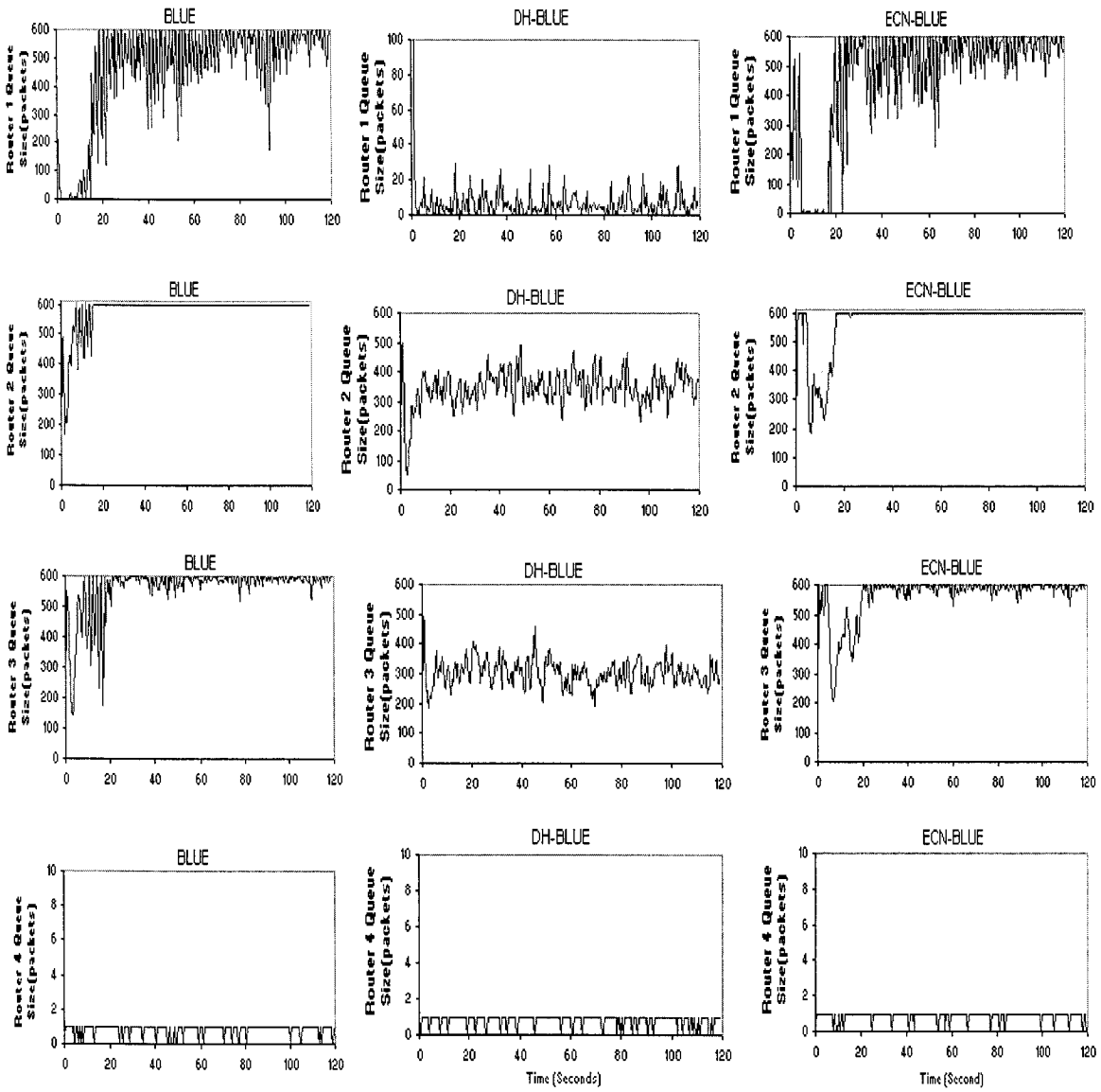

Figure 5-7 Instantaneous Queue Size Comparison of BLUE, DH-BLUE, with ECN-BLUE with the data service rate of 45-45-45-45 in Scenario of 700-FTP

Figure 5-8 shows the average packet drop rates of RED, DH-RED and ECN-RED with the data service rate of 45-45-45-45 in Scenario C. We can see that all three algorithms present the equally low packet drop rates in Router 1, the highest drop rates in Routers 2 and 3, and no drop in Router 4. We further observe that, the packet drop rate of DH-RED is higher than RED, a little unlike the situation in the single bottleneck network. This difference comes from the different combinations of router data service rates, i.e., 45-45-45-45 in

Scenario C versus 45-infinity-infinity-infinity in the case of the single bottleneck network, which is also a factor that affects the performance of queue size in a certain router, and the different traffic loads. However, the packet drop rate of ECN-RED is much lower than both of RED and DH-RED due to the packet marking strategy.

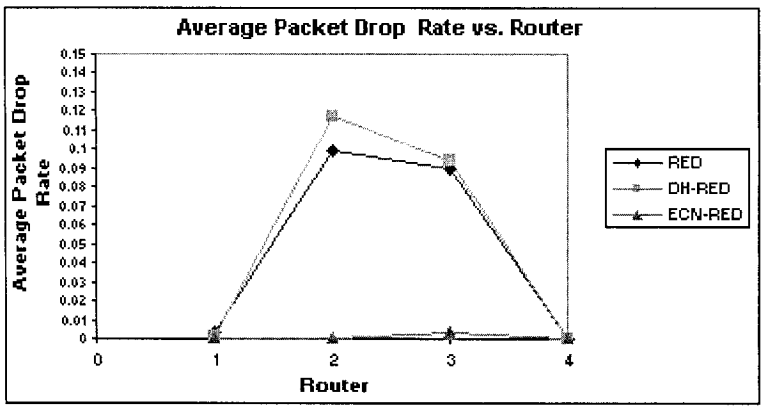

Figure 5-8 Average Packet Drop Rate of RED-related Algorithms versus Router with Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

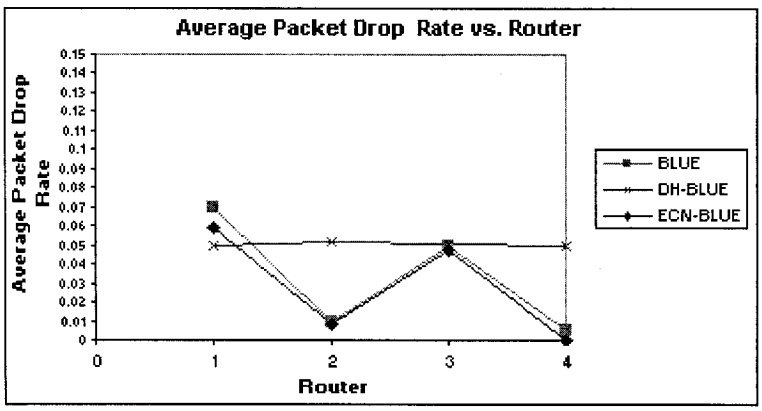

Figure 5-9 Average Packet Drop Rate of BLUE-related Algorithms versus Router with Data Service Rate of 45-45-45-45 in Scenario of 700-FTP

Figure 5-9 shows the average packet drop rates of BLUE, DH-BLUE and ECN-BLUE with the data service rate of 45-45-45-45 in Scenario C. As can been seen from Figure 5-9, DH-BLUE presents higher packet drop rates than BLUE in all routers except for Router 1. The average packet drop rate of ECN-BLUE is similar to BLUE, which is generally lower than DH-BLUE.

5.4.2 Scenario D (200 FTP sources and 500 HTTP sources)

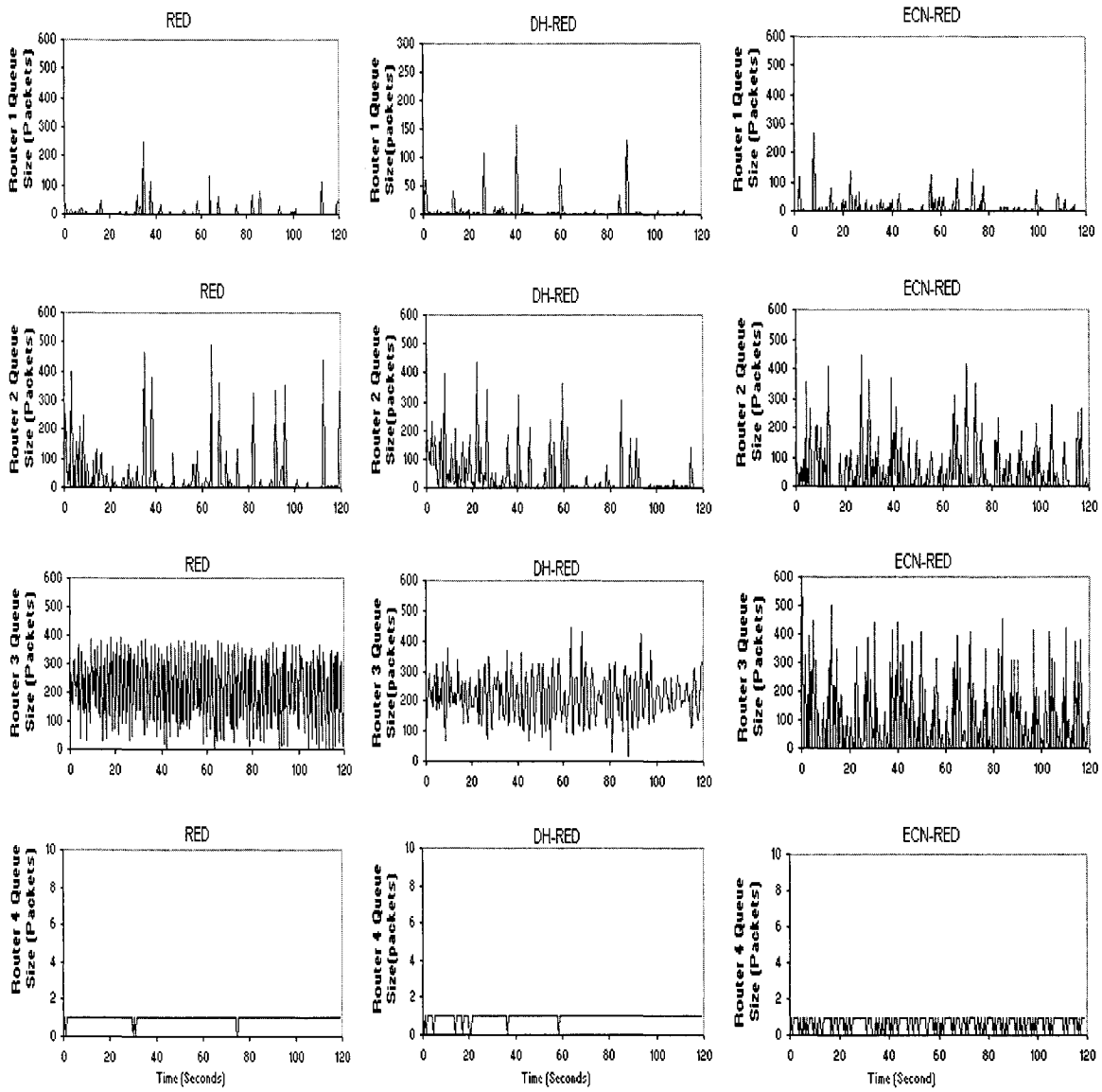

Figure 5-10 Instantaneous Queue Size Comparison of RED, DH-RED with ECN-RED with Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

Figure 5-10 shows the instantaneous queue sizes of RED, DH-RED, and ECN-RED, with the data service rate of 45-45-45-45 in Scenario D (200 FTP sources and 500 HTTP sources). From Figure 5-10, we can see that DH-RED only presents a little stable queue size when compared with RED, unlike the situation in Scenario C. This is because the traffic load in Scenario D is much lighter than Scenario C, which makes the effect of head dropping

strategy insignificant. The queue size fluctuation of ECN-RED exhibits similarly to Both DH-RED and RED.

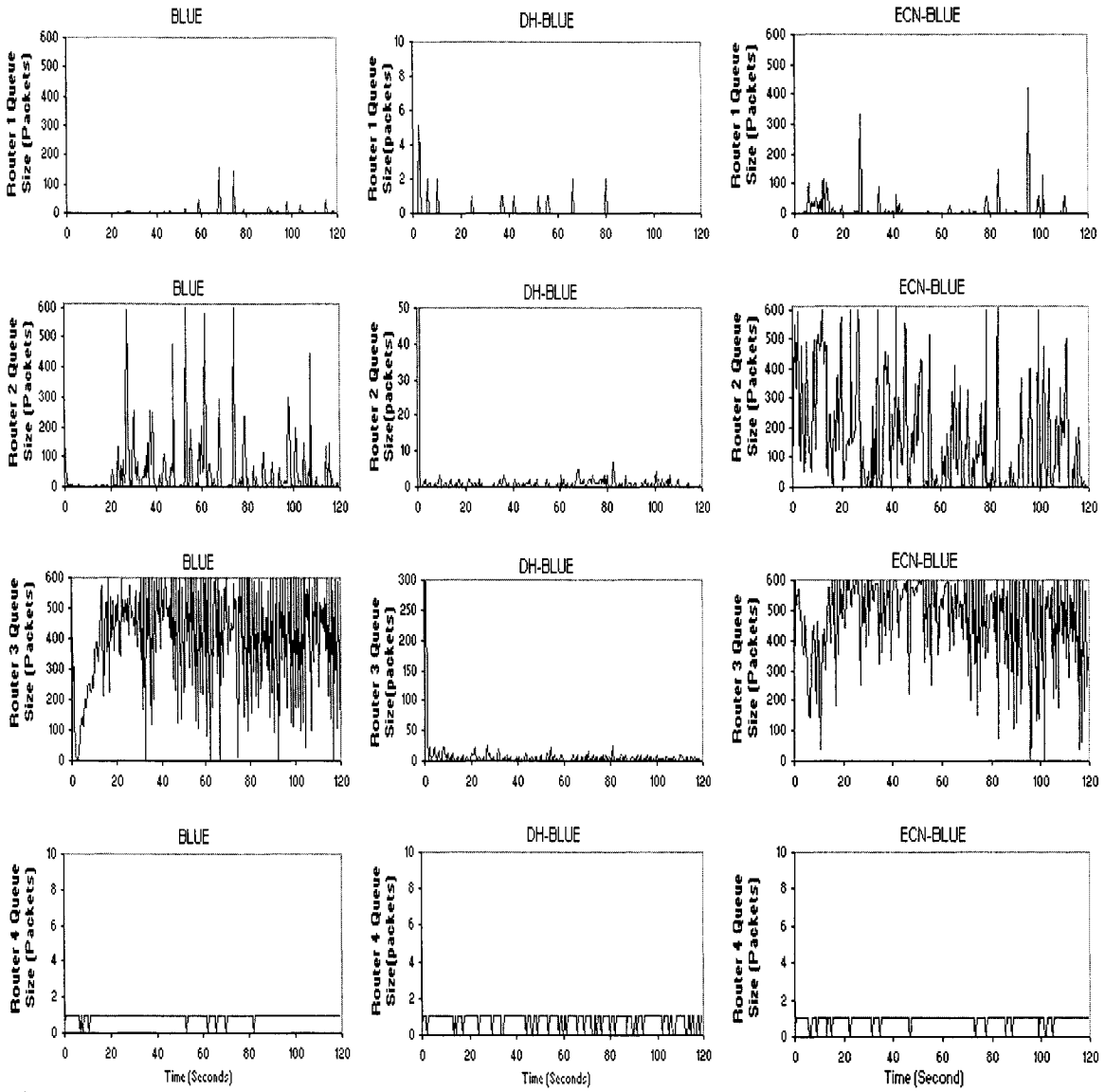

Figure 5-11 Instantaneous Queue Size Comparison of BLUE, DH-BLUE with ECN-BLUE with Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

Figure 5-11 shows the instantaneous queue sizes of BLUE, DH-BLUE, and ECN-BLUE with the data service rate of 45-45-45-45 in Scenario D. As can be seen that DH-BLUE presents a relatively stable queue when compared with BLUE (even it exhibits a little higher initial queue size for the same reason as previously mentioned). Like in Scenario C, ECN-BLUE oscillates as much as BLUE.

Figure 5-12 shows the average packet drop rates of RED, DH-RED and ECN-RED with the data service rate of 45-45-45-45 in Scenario D. As can be see from Figure 12 that DH-RED presents a lower packet drop rate than RED, and the drop rate of ECN-RED is much lower than both of RED and DH-RED.

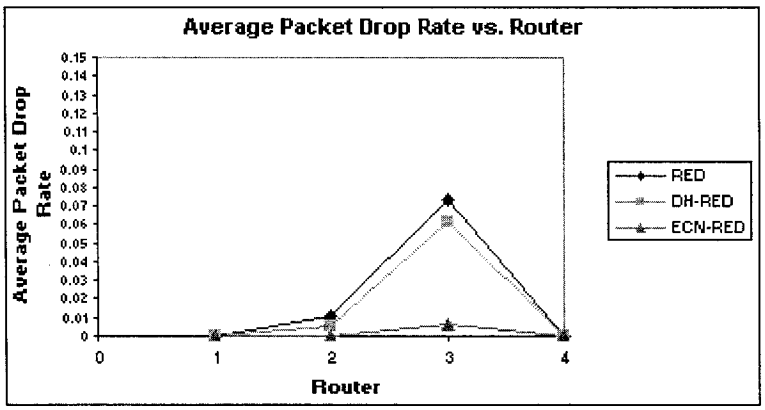

Figure 5-12 Average Packet Drop Rate of RED-related Algorithms versus Router with Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

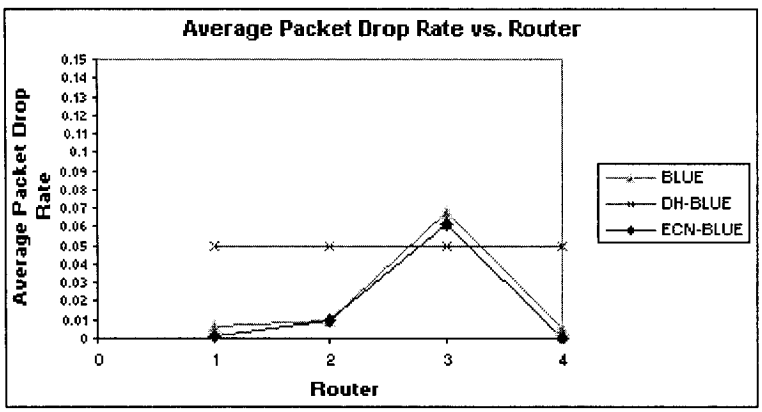

Figure 5-13 Average Packet Drop Rate of BLUE-related Algorithms versus Router with Data Service Rate of 45-45-45-45 in Scenario of 200-FTP-plus-500-HTTP

Figure 5-13 shows the average packet drop rates of BLUE, DH-BLUE, and ECN-BLUE with the data service rate of 45-45-45-45 in Scenario D. We can see that DH-BLUE presents higher packet drop rate than BLUE in all routers except for Router 3, and ECN-BLUE behaves almost the same as BLUE.

5.5 Concluding Remarks

Generally, DH-AQM techniques can greatly stabilize their queue size when compared with AQM techniques, except that DH-BLUE exhibits a relatively large queue size in the beginning of simulation. However, its average packet drop rate present differently. The effect of head dropping on the performance of queue size stability generally depends on how much the percentage of the queuing delay occupies in the total round trip delay.

Chapter 6 Conclusions

In this thesis, we have proposed and evaluated some ECN and head dropping techniques to solve the problems of queue size oscillation and unnecessary packet loss experienced by RED (the only deployable AQM algorithm in current networks) and some other AQM algorithms.

We have examined the effect of the ECN mechanism on AQM techniques by comparing ECN-RED, ECN-BLUE, ECN-ARED and ECN-PI-RED with their corresponding AQM algorithms, RED, BLUE, ARED and PI-RED, under a wide verity of situations in the two different network environments, the single bottleneck network and the multi-bottleneck network. The investigation work based on the latter network gives more practical observations to network performance. Instead of discarding packets as AQM algorithms do, ECN-AQM schemes use the method of marking packets to inform TCP sources of congestion when congestion has been detected. Numerous simulation results have shown that our investigated ECN-AQM algorithms, except for ECN-BLUE, can well improve the problem of unnecessary packet loss, one of the major problems encountered by most AQM algorithms, reduce the average queue size of AQM in most cases, but make queue size oscillation even worse. Obviously, ECN-AQM schemes are not suitable to be used in the network with the involvement of multimedia traffic. While ECN–BLUE cannot further improve the performance of BLUE in terms of packet drop rate, average queue size and queue size stability. We have also found that ARED and ECN-ARED cannot respectively outperform RED and ECN-RED on these performance measurements. Our simulations have also revealed that the combination of ECN and PI-RED cannot solve both of the problems experienced by RED. Our simulation experience regarding the performance of RED with some different combinations of router data service rates indicates that a RED router would have a relatively larger queue size and a higher packet drop rate (given a fixed traffic load in the router), as the router data service rate is lowered.

Aiming at solving the performance problem of queue size oscillation, another major shortcoming of most AQM algorithms, we have investigated a head dropping strategy to be applied to some AQM algorithms such as RED and BLUE, in order to decrease the delay of

the transmission of congestion information occurring in the bottleneck router up to a time period of the queuing delay earlier to consequently improve the performances. By comparing DH-RED and DH-BLUE with RED and BLUE, we found that both DH-RED and DH-BLUE can help the original RED and BLUE algorithms achieve a more stable queue size.

6.1 Future Work

There are still many remaining issues to be explored regarding the performance of congestion control algorithms in the multi-bottleneck network, e.g., how the other algorithms such as BLUE (not only RED presented in this thesis) perform as the router data service rates change. In addition, we are also interested in study the network performance when other types of traffic such as UDP traffic appear in the network, and when ACKs generated from the destination also experience being lost in the network.

References

[AlPa99] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", *RFC 2581*, April 1999. HTTP://www.faqs.org/rfcs/rfc2581.html.

[ArZh02] R. M. Arora, H. Zhang, and O. Yang, "TCP Network with ECN over AQM", OPNETWORK2001, Washington DC, August 2002.

[BaPa97] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performances over Wireless Links", *IEEE/ACM Transactions on Networking,* Vol. 5, pp.756-769, December 1997.

[BoMa00] T. Bonald, M. May, and J-C.Bolot, " Analytic Evaluation of RED Performance", *Proc. IEEE Infocom*, March 2000.

[BrOM94] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance", *Proceedings of ACM SIGCOMM 1994*, pp. 24-35, London, UK, May 1994.

[ChJe00] M. Christiansen, K. Jeffay, D. Otta, and F.D. Smith, "Tuning RED for web traffic", *Proc. IEEE/ACM Transaction on Networking*, Vol.9, No.3, pp. 249-264, June 2000.

[CoYa04] J. Cong, O. Yang, H. Zhang, "Enhancing the TCP Traffic Control Performance with A Proportional and Integral rate (PIR) controller", in preceding of *ICC* 2004, Paris, France, June 2004.

[CoZh03] J. Cong, H. Zhang and O. Yang, "A Queuing Performance Comparison of Extended Proportional Controller in TCP Traffic Control", *Proceedings.of IEEE CCECE 2003,* pp. 973-976, Montreal, Canada, April 2003.

[CrBe97] M. Crovella and A. Bestavros, "Self Similarity in World Wide Web Traffic: Evidence and Possible Causes", *IEEE/ACM Transactions on Networking,* Vol. 5, pp.835-846, December 1997.

[FaFl96] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communication Review*, Vol. 26, No. 3, pp. 5-21, July 1996.

[FeKa99] W. Feng, D. D. Kandlur, D. Saha and K.G. Shin, "A Self-configuring RED Gateway", *Proceedings of IEEE Infocom 1999*, pp. 1320-1328, New York, NY, March 1999.

[FeKa02] W.C. Feng, D.D. Kandlur, D. Saha, K.G. Shin, "The BLUE active queue management algorithms", *IEEE/ACM transaction on Networking*, Vol. 10, No.4, pp. 513-528, August 2002.

[FiBo00] V. Firoiu and M. Borden, "A Study of Active Queue Management for Congestion Control", in *Proceedings of IEEE Infocom 2000*, pp. 1435--1444, Tel-Aviv, Israel, March 2000.

[FlHe99] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", *RFC 2582*, April 1999. HTTP://www.faqs.org/rfcs/rfc2582.html.

[FlGu01] S. Floyd, R. Gummadi and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management", *Technical Report*, August 1, 2001.

[FlJa93] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Control", *IEEE ACM Transactions on Networking*, Vol. 1, No. 4, pp.397-413, 1993.

[Floy94] S. Floyd, "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, Vol. 24, No. 5, pp. 10-23, October 1994.

[GeWe00] M. Gerla, W. Weng and R. L. Cigno, "Bandwidth Feedback Control of TCP and Real Time Sources in the Internet", *Proceedings of IEEE Globecom 2000*, pp. 561--565. San Francisco, CA, USA, November 2000.

[Hash89] E. Hashem, "Analysis of random drop for gateway congestion control" Report LCS TR-465, Laboratory for Computer Science, MIT, Cambridge, MA, 1989.

[Hoe96] J. C. Hoe, "Improving the Start-up Behavior of a Congestion Control Scheme for TCP ", *Proceedings of ACM Sigcomm 1996*, pp. 270-280, Stanford, CA, August 1996.

[HoMi01a] C.V. Hollot, V. Misra, D. Towsley and W. Gong, "A Control Theoretic Analysis of RED", *Proceedings of IEEE Infocom 2001*, pp. 1510-1519, Anchorage, AK, April 2001.
HTTP://www-et.cs.umass.edu/papers/papers.html.

[HoMi01b] C.V. Hollot, V. Misra, D. Towsley and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows", *Proceedings of IEEE Infocom 2001*, pp. 1726-1734, Anchorage, Alaska, USA, April 22--26, 2001.

[Jaco88] V. Jacobson, "Congestion Avoidance and Control", *Proceedings of ACM Sigcomm 1988*, pp. 314-329, Stanford, CA., August 1988.

[Jaco90] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm", message to end2end-interest mailing list, April 1990. 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.

[KaKa00] S. Karandikar, S. Kalyanaraman, P. Bagal and B. Packer, "TCP Rate Control", *Computer Communications Review*, Vol. 30, No. 1, pp. 45-58, January 2000.

[KaKu00] A. Karnik and A. Kumar, "Performance of TCP Congestion Control with Explicit Rate Feedback: Rate Adaptive TCP (RATCP)", *Proceedings of IEEE Globecom 2000*, San Francisco, CA, USA, November 2000.

[KuLa01] P. Kuusela, P. Lassila and J. Virtamo, "Stability of TCP-RED Congestion Control", *Proceedings of ITC-17*, pp. 655-666, Salvador da Bahia, Brazil, 2001.

[LaNe96] T.V. Lakshman, A. Neidhardt, T.J. Ott, "The Drop from Front Strategy in TCP and in TCP over ATM", *Infocom96*, pp. 1242-1250, 1996.

[LeTa94] W.E. Leland, M.S. Taqqu, W. Willinger and D.V. Wilson, "On the self-similar Nature of Ethernet Traffic (extended version)", *IEEE ACM Transactions on Networking*, Vol. 2, No. 1, pp.1-15, 1994.

[LoPa02] S.H. Low, F. Paganini, J. Wang, S. Adlakha, J.C. Doyle, "Dynamics of TCP/RED and a scalable control", *Proceedings of IEEE Infocom 2002*, New York USA, June 2002.

[MaMa96a] M. Mathis and J. Mahdavi," Forward Acknowledgment: Refining TCP Congestion Control ", *Computer Communication Review*, Vol. 26, No. 4, pp. 281-291, October 1996.

[MaMa96b] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options", *IETF RFC2018,* October 1996. HTTP://www.faqs.org/rfcs/rfc2018.html.

[Masc99] S. Mascolo, "Congestion Control in High-Speed Communication Networks Using the Smith Principle", *Automatica Journal, Special Issue on "Controls Methods for Communications Networks"*, Vol.35, No.12, pp.1921-1935, December 1999.

[MaSe97] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoid Algorithm", *ACM Computer Communication Review,* Vol. 27, No. 3, pp. 67-82, July 1997.

[Morr97] R. Morris, "TCP Behavior with Many Flows", *IEEE International Conference on Network Protocol*, October 1997.

[Morr00] R. Morris, "Scalable TCP congestion control", in *Proceedings of IEEE Infocom, Tel Aviv*, pp. 1176-1183, March 2000.

[Nagl84] J. Nagle, "Congestion Control in IP/TCP Internetworks", *RFC 896*, FACC Palo Alto, 6 January 1984. HTTP://www.faqs.org/rfcs/rfc896.html.

[OPNE00] "OPNET Modeler Manuals", OPNET Version 7.0, OPNET Technologies, Inc, 2000.

[OtLa99] F.J. Ott, T.V. Lakshman and L. Wong, "SRED: Stabilized RED", *Proceedings of IEEE Infocom 1999*, pp 1346-1355, Piscataway, N.J., 1999.

[RaFl99] K. Ramakrishnan and S. Floyd, " A Proposal to add Explicit Congestion Notification (ECN) in IP", *IETF RFC 2481,* January 1999.

[SaAh00] J. Hadi Salim and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks', *IETF RFC 2884*, July 2000.

[Sast75] A.R.K. Sastry, "Improving repeat-request (ARQ) performance on satellite channels under high error rate conditions", *IEEE Transactions on communications*, Vol. Com-23, No. 4, pp.436-439, April 1975.

[Stev94] W. Stevens, *TCP/IP* Illustrated, Volume 1, Addison-Wesley, Reading, MA, 1994.

[Stev97] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *RFC 2001*, January 1997. HTTP://www.faqs.org/rfcs/rfc2001.html.

[TiMa01] P. Tinnakornsrisuphap and A. M. Makowski, "Queue Dynamics of RED Gateways under Large Number of TCP Flows", *Proceedings of IEEE Globecom 2001*. San Antonio, Texas, USA, November 2001.

[YiHl93] N. Yin, and M.G. Hluchyj, , "Implication of dropping packets from the front of a queue,", *IEEE Trans. Commun.,* Vol. 41, pp. 846-851, June 1993.

[ZhYa97] H. Zhang, O. Yang and H. Mouftah, "Design of Robust Congestion Controllers for ATM Networks", *Proceedings of IEEE Infocom 1997*, pp. 302-309, Kobe, Japan, April 1997.

[ZhYa00] H. Zhang, M. Yang, O. Yang and H. Mouftah, "Design of Proportional Congestion Control for High-speed Networks with Variable Delays", *Proceedings of IEEE ICC 2000*, pp. 1435-1439, New Orleans, USA, June 2000.

Appendix A AQM Algorithms

We summarize all of the studied AQM algorithms in the following sections.

A.1 RED

RED (Random Early Detection) was originally proposed by Floyd [FlJa93] to solve the severe performance problems experienced by the early gateway queue management algorithms, e.g. the global synchronization and the bias against bursty traffic [FlJa93].

The RED gateway uses a low-pass filter with an exponential weighted moving average to calculate the average queue size avg, which measures the traffic load in networks, according to the following Equation (A1-1).

$$avg = (1 - w_q) \bullet avg + w_q \bullet q, \qquad \text{(A1-1)}$$

where w_q is queue weight and q is current queue size. The average queue size avg is then compared with two preset parameters, the minimum threshold $th_{\min}$ and the maximum threshold $th_{\max}$. If avg is less than $th_{\min}$, no packets will be dropped. If it is larger than $th_{\max}$, then all incoming packet will be dropped. If avg is between $th_{\min}$ and $th_{\max}$, the incoming packet will be dropped with a probability p_a, which is a function of the average queue size, and is calculated based on Equation (A1-2), from which we can see that RED ensures that the router does not wait too long before dropping a packet.

$$p_a = p_b / (1 - count \cdot p_b) \quad , \qquad \text{(A1-2)}$$

where $count$ is the number of packets that have been forwarded since the last dropped packet and p_b is the temporary packet-dropping probability calculated in Equation (A1-3), which linearly increases as avg increases when avg is between the $th_{\min}$ and $th_{\max}$, like being indicated in Figure A1-1.

$$p_b = p_{\max}(avg - th_{\min}) / (th_{\max} - th_{\min}) \quad , \qquad \text{(A1-3)}$$

where $p_{\max}$ is the maximum allowed value for p_b.

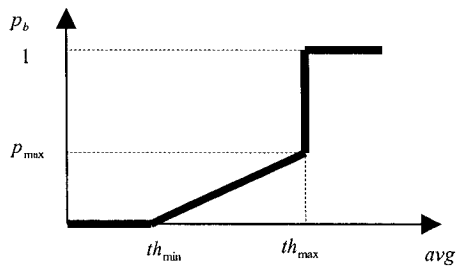

Figure A1-1 Temporary Packet-dropping Probability of RED

Figure A1-2 shows the RED algorithm, which exactly reflects the main idea of RED we have stated before. In addition, we can see that the RED algorithm contains two parts. One is the determination of the degree of burstiness allowed in the router queue by the computation of average queue size, which is the basis of the different congestion control actions taken later. Another is the determination of how frequently an incoming packet should be dropped according to the current congestion level, through the calculation of packet-dropping/marking probability, which ensures that the router drop/mark a packet at fairly evenly-spaced intervals and sufficiently frequently at the same time to control the average queue size, in order to avoid global synchronization and biases.

A.2 BLUE

BLUE was originally proposed by Feng et al. [FeKa02] for solving the unnecessary packet loss problem suffered by RED and some other AQM techniques. It directly uses packet loss and link idle, instead of relying on the instantaneous or average queue size as congestion indicator to manage congestion. BLUE has been shown to perform significantly better than RED in terms of both packet loss rate and buffer size requirement in networks.

Figure A2-1 shows the BLUE algorithm. As can be seen from the algorithm, upon arrival of a packet, BLUE maintains a single probability, *p(n),* which it uses to mark (or drop) the packet when it is queued at the tail of the queue. If the queue length continually exceeds a certain value or the queue has been discarding packets due to a buffer flow for a given period of time (called freeze time, which is used to determine the minimum time interval for updating a packet-dropping probability), the packet drop probability will be increased by a constant factor *d1*, increasing the rate at which it sends back congestion notification. Conversely, if the link remains idle or the queue becomes empty for the freeze time, then the

drop probability will be decreased by another constant factor *d2*. Such a mechanism allows BLUE to effectively correct the transmission rate it needs to send back a congestion notification.

```
Initialization:
        avg := 0
        count := -1
for each packet arrival
        calculate the new average queue size avg:
                if the queue is nonempty
                        avg := (1 - w_q)avg + w_q q
                else
                        m := f(time - q_time)
                        avg := (1 - w_q)^m avg
        if  th_min ≤ avg < th_max
                count++
                calculate  p_a
                        p_b := p_max (avg - th_min)/(th_max - th_min)
                        p_a := p_b /(1 - count · p_b)
                drop the incoming packet with  p_a
                count := 0
        else if  th_max ≤ avg
                drop the incoming packet
                count := 0
        else
                count := -1
when queue becomes empty
        q_time := time

Saved Variables:
        avg :       average queue size
        q_time :    queue idle start time
        count :     number of packets since last
                                        \dropped packet
Parameters:
        w_q :    queue weight
        th_min :  minimum threshold for queue
        th_max :  maximum threshold for queue
        p_max :  maximum allowed value for  p_b
Other:
        p_a :    current packet-dropping probability
        q :      current queue size
        time :   current time
        f(t) :   linear function of the time  t
```

Figure A1-2 Algorithm of RED

```
Upon packet loss (or q(n)>B) event:
      if ((now- last_update)> freeze_time) then
                  p(n) = p(n-1) + d1
                  last_update=now
Upon link idle event:
      if ((now- last_update)> freeze_time) then
                  p(n) = p(n-1) - d2
                  last_update=now

Saved Variables:
      q(n):     average queue size
      last_update:  time for last updated
      p(n):     packet drop probability

Parameters:
      freeze_time:       freeze time
      d1:    drop probability increment value
      d2:    drop probability decrement value
      B:     buffer size
Other:
      now:  current time
```

Figure A2-1 Algorithm of BLUE

Also we can see from Figure A2-1 that except for the parameter of *p(n),* BLUE uses other two parameters, *freeze-time* and parameter pair *d1* and *d2*, to determine how quickly the dropping/marking probability changes over time. In addition, we notice that the dropping/marking probability under the situation that the queue length exceeds a certain value is treated in the same way as that in response to packet losses. This permits space to be left in the queue for transient traffic bursts, and allows the queue to control queuing delay when the length of queue being used is large.

A.3 ARED

ARED (Adaptive RED) was originally proposed by Feng et al. [FeKa99]. However, when people talk about the ARED algorithm, they generally mean the modified algorithm to the original one, which was proposed by Sally Floyd [FlGu01]. This revised version was developed by the authors after recognizing that the average queue size in RED is quite sensitive to the level of congestion and to the setting of RED parameters, and therefore not predictable. They made minimal changes to the original RED algorithm to remove the sensitivity to the parameters that effect the performance of RED, configure its parameters

automatically in response to the traffic load status, and eventually achieve a pre-defined target queue size in a wide variety of traffic scenarios, without sacrificing the other advantages of RED.

Figure A3-1 shows the ARED algorithm. We can see that the maximum packet drop probability of ARED will be additive-increased or multiplicative-decreased by two different factors according to the traffic load status, in order to keep the average queue size not just between the maximum threshold $th_{\max}$ and the minimum threshold $th_{\min}$, but within a queue size target range half way between these two thresholds. Observe also that the maximum packet drop probability $p_{\max}$ is adjusted slowly and infrequently, which guarantees the robustness of ARED, and is constrained to remain within the range [0.01,0.5] in order to ensure that the overall performance of RED should be acceptable during the transition period even though the average queue length might not be in its target range, and the average delay or throughput might suffer slightly.

Every *interval* seconds:
    if *avg* > *target* and $p_{\max}$ <= 0.5)
        increase $p_{\max}$:
        $p_{\max} = p_{\max} + \alpha_{ARED}$;
    elseif (*avg* < *target* and $p_{\max}$ >= 0.01)
        decrease $p_{\max}$:
        $p_{\max} = p_{\max} * \beta_{ARED}$;

Saved Variables:
    avg : average queue size
    $p_{\max}$: maximum packet drop probability

Parameters:
    interval: time; 0.5 seconds
    target: target for *avg* ; [$th_{\min}$ + 0.4 * ($th_{\max}$ - $th_{\min}$), $th_{\min}$:+0.6 * ($th_{\max}$ - $th_{\min}$)].
    α_{ARED} : increase factor; min (0.01, $p_{\max}$ /4)
    β_{ARED} : decrease factor; 0.9

Figure A3-1 Algorithm of ARED

A.4 PI-RED

Even though RED is the only deployable AQM algorithm advocated by IETF, it has a serious queue size oscillation problem. To overcome this drawback, network researchers have proposed many RED variants to stabilize the queue size and maintain it at a desired level through different ways. After realizing two limitations of RED, the tradeoff between response speed and queue size stability and the direct coupling of queue length and loss probability, C.V. Hollot et al. proposed the PI-RED (Proportional and Integral controller in RED capable routers) [HoMi01b]. PI-RED is shown to outperform RED significantly in terms of a high level of utilization and a low level of latency.

Figure A4-1 shows the implementation of the PI controller in a RED capable router where p is the packet drop probability assigned to the incoming packet, p_0 is the reference dropping probability, δp is the PI controller output, q is the router queue size, q_0 is the reference queue size (desired queue size level), and δq is the queue size regulation error, which is the input of PI controller. The PI controller is designed by applying the classical control theory techniques to determine the packet drop probability and control the averaging queue size at a reference level. As shown in Figure A4-1, PI-RED, Unlike RED, does not determine the packet drop probability directly from the queue length, but uses the computed output of PI controller to determine its packet drop probability.

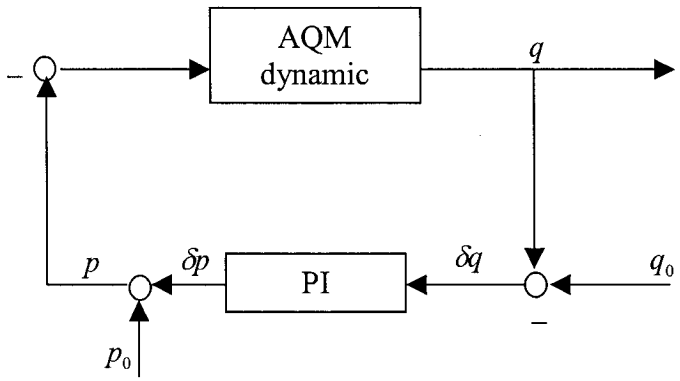

Figure A4-1 Implementation of PI Controller in RED Capable Router

for each packet arrival
    if $q \geq$ physical memory limit
        drop the packet
    else
        drop the packet with probability p

for each probability sampling time
    $p := a \cdot (q - q_0) - b \cdot (q_{old} - q_0) + p_{old}$
    $p_{old} := p$
    $q_{old} := q$

Saved Variables:

q : current queue size

q_{old} : old queue size

p : current dropping probability

p_{old} : old dropping probability

Parameters:

q_0 : reference queue size

a, b : PI controller constants

Figure A4-2 Algorithm of PI-RED

Figure A4-2 shows the PI-RED algorithm. As can be seen that its implementation requires a modification to the averaging algorithm in RED, and keeps two additional state variables, which are the instantaneous queue length at the previous sampling time and the packet drop probability at the previous sampling time. For each probability sampling time, if there is a difference between the average queue size and the reference queue size, then a PI controller will verify its output to change the packet drop probability accordingly.

A.5 ECN

The ECN (Explicit Congestion Notification) algorithm studied in this thesis was proposed by Sally Floyd [Floy94]. This algorithm is the one that people mean the ECN mechanism and pay much attention to, even though there are some other ECN mechanisms before this algorithm such as Source Quench messages and DEC bit's ECN bit. In contrast to these two schemes, ECN [Floy94] uses a different mechanism to solve the problem of unnecessary packet loss observed in RED, and therefore avoid unnecessary delay for a packet. The basic mechanism of ECN is summarized as follows:

In the TCP connection setup phase, both TCP source and destination exchange information about their willingness to use ECN. Once an agreement to use ECN is reached, the TCP source tells the router that the end nodes are ECN-capable by setting the ECN-capable Transport (ECT) bit in its IP header of the packet. The router then sets the Congestion Experienced (CE) bit in the IP header of the incoming packet if congestion is detected, instead of dropping the packet, and next forwards the packet to the destination. In turn, the destination sets the ECN-echo (ECE) flag in the TCP header of the next outgoing TCP ACK being about to send to the source when receiving the packet with the CE bit set. It will continue to do so in the subsequent ACKs until receiving from the source an indication that the source has responded to the congestion notification, and send them back to the source.

Upon receipt of a TCP ACK with ECE flag set, the TCP source triggers the congestion avoidance algorithm by halving its congestion window, *cwnd* and its congestion window threshold value, *ssthresh*, to therefore reduce its transmission rate in response to the ACK with the CE bit set. These actions are the same as those in response to a single packet loss. After the appropriate steps for congestion control have been taken by the source, it sets the Congestion Window Reduced (CWR) bit on the next outgoing packet to acknowledge the destination that it has reacted to the notification of congestion (the ECE flag). Note that the TCP congestion control mechanism (TCP-Reno in this thesis) still works in response to three duplicate ACKs (indicating packet loss). However, the response to an ECN does not trigger the sending of any new or retransmitted packets. On the other hand, if the agreement is not be reached, TCP-Reno function as usual, that is, the source halves the *cwnd* and reduces the *ssthresh* when packet losses are detected. From the above description, we have noticed that ECN needs the cooperation from both routers and TCP end hosts (sources and destinations).

Some extra bits in the TCP and IP Header of the packet are required to support the ECN mechanism. In the IP Header, there are two bits used for ECT and CE, respectively. The ECT bit is set by the source if both of the end hosts are ECN capable, and the CE bit is set by the router if congestion is detected. In the reserved field of TCP Header, there are another two bits, ECE flag and CWR flag, used by the source and destination when they negotiate the desire and the capability to use ECN, and take the subsequent actions in case there is congestion experienced in the network.

AQM techniques such as RED, BLUE usually use ECN as their alternative congestion indication policy to determine when an arriving packet will be marked for congestion instead of dropping packets.

```
for each new ACK arrival with ECN bit set
      if no ECN reaction has been done and no dropped
      \packet reaction has been done in the last
      \round-trip time
            cwnd := cwnd / 2
            ssthresh := ssthresh / 2
      else
            do normal Reno reaction to new ACKs

Saved Variables:
      cwnd:       congestion window size
      ssthresh:   slow start threshold
```

Figure A5-1 Part of Algorithm of Reno-ECN

Figure A5-1 shows the part of the Reno-ECN algorithm suggested by [Floy94]. We have noticed from the algorithm that TCP's response to ECN is similar to its response to a dropped packet (duplicate ACKs). Unlike its response to duplicate ACKs that at least three duplicate ACKs can trigger a response to congestion, the receipt of a single ECN should trigger a response to congestion, e.g. halving the *cwnd* size and *ssthresh* in the case of TCP-Reno. However, TCP ignores the succeeding ECNs if it has reacted to a previous ECN or a dropped packet until all outstanding packets have been ACKed.