Université d'Ottawa • University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

CONG, Jun

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Enhancing TCP Traffic Flow Performance with a Proportional and Integral Rate Controller: Theory, Design, and Performance Evaluation

Oliver Yang

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

P.X. Liu

H. Mouftah

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Enhancing TCP Traffic Flow Performance with A Proportional and Integral Rate Controller: Theory, Design, and Performance Evaluation

By

Jun Cong

A thesis submitted to
School of Graduate Studies and Research
in partial fulfillment of requirements for the degree of

Master of Applied Science

Master Program in Electrical Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

January 15, 2004

Your file Votre référence
ISBN: 0-494-01450-4
Our file Notre référence
ISBN: 0-494-01450-4

Abstract

This thesis proposes a new network traffic congestion control algorithm – the PIR (Proportional and Integral Rate) controller. It is designed to overcome the shortcomings of RED and some other related AQM (Active Queue Management) techniques developed in recent years. That is to avoid unnecessary packet losses and stabilize the queue length in the router. It can also prevent the throughput reduction in the presence of wireless links. We proved the stability of the PIR controller theoretically and conducted the performance analysis and comparison for the PIR controller and some other common traffic congestion control schemes.

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Oliver Yang, for his research guidance and suggestions throughout the research. I also wish to extend my thanks to Dr. Hongyi Zhang for his kind help.

I would like to express my deep gratitude to my family, especially my wife Xintong 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	1.2.2
ATM	Asynchronous Transfer Mode	1.2.3
DH	Drop Head	1.2.2
DRED	Dynamic RED	1.2.2
DT	Drop Tail	1.2.2
ECN	Explicit Congestion Notification	1.2.3
FIFO	First in First out	1.2.2
FRED	Flow RED	1.2.2
FSM	Finite State Machine	2.4.0
FTP	File Transfer Protocol	2.3.2
HTTP	Hypertext Transfer Protocol	2.3.1
PID	Proportional, Integral and Differential	1.5.0
PIR	Proportional & Integral Rate Controller	1.2.2
PIR-A	Implementation Method A of PIR Controller	2.1.0
PIR-B	Implementation Method B of PIR Controller	2.1.0
PI-RED	Proportional & Integral Controller in RED Router	1.2.2
QoS	Quality of Service	1.2.0
RED	Random Early Detection	1.1.0
RTT	Round-Trip Time	2.1.1
SRED	Stabilized RED	1.2.2
TCP/IP	Transmission Control Protocol / Internet Protocol	1.1.0

List of Notations and Symbols

		Section of 1st appearance
$a_i(t)$	Indicator. $a_i(t)=1$ if Source i is active. Otherwise, $a_i(t)=0$.	2.2.0
d	PIR controller adaptation constant	3.1.0
$e(t)$	Queue size error. $e(t)=q(t)-q_{SP}$	2.2.0
$f_i(t)$	Transmission rate of TCP source i	2.2.0
k	Global proportional gain	3.1.0
k_i	Proportional gain for Source i	2.2.0
p_h	Parameter of think time probability distribution	2.3.1
p_l	Parameter of think time probability distribution	2.3.1
$q(t)$	Real-time queue size in the router	2.2.0
q_{SP}	Reference queue size in the router	2.2.0
$q_{sp}(t)$	Setpoint of queue length $q(t)$	3.1.0
$rtt_i(t)$	Round-trip delay of TCP connection i	3.3.1
s	Bottleneck router service rate	2.2.0
$w(t)$	Rate of non-TCP-controlled source packets entering the router	2.2.0
$win_i(t)$	Window size of Source i	3.3.1
$win_i^{adv}(t)$	Router-advertised window size for Source i	3.3.1
$x(t)$	State space matrix of time-delay control system	3.2.0
A	Transformation matrix of control system	3.2.0
C	Transformation matrix of control system	3.2.1
N	Number of TCP connections in the network	2.2.0
R	Positive-definite symmetric matrix	3.2.1
RTT	Pre-set round-trip delay for PIR-B	3.3.2
$V(x(t))$	Lypunov function	3.2.1
α	Parameter of file length probability distribution	2.3.1
β	Positive constant that satisfies $0 \le \beta \le 1$	3.2.1
η_h	Parameter of think time probability distribution	2.3.1
η_l	Parameter of think time probability distribution	2.3.1
$\mu(t)$	Global recommended source rate	3.1.0
$\mu_i^{\max}$	Peak transmission rate recommended for TCP source i	2.2.0
$\mu_{\max}$	Global recommended peak source rate	3.1.0
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
τ_i	Round-trip delay for Source i	2.2.0
τ_{i1}	Path delay from Source i to the bottleneck router	2.2.0
τ_{i2}	Path delay from the bottleneck router back to Source i through the destination	2.2.0
τ_{i3}	Path delay from the router to Source i	3.3.2
τ_{max}	Maximum round-trip delay	3.2.2

Chapter 1
Introduction

1.1 Overview

Due to rapid technological development, computers have made spectacular progress in a short time, compared to other industries. The merging of computers and communications had a profound influence on the way computer systems are organized. In the 1970s and early 1980s, many companies have operated in the computer network environment. Since the 1990s, computer networks have begun to deliver Internet services to private individuals at home. The boom of internetworking has required high efficient traffic engineering mechanisms to manage the traffic exposed to the network, and is stressing network traffic and resources for congestion control implementation. As the dominant network protocol suite, TCP/IP (Transmission Control Protocol / Internet Protocol) plays a very important role in the network traffic control. Its Reno algorithm [Jaco90] has been the most-adopted traffic control algorithm in computers. On the other hand, Active Queue Management (AQM) is a very active research area in networking. RED (Random Early Detection)[1] [FlJa93], the most popular AQM algorithm, works in the router to cooperate with the Reno algorithm working in the source and destination to ensure that congestion is avoided.

The queue size stability in the router is very important to the network performance. Without the stability of queue size, more packets will be dropped, the time delay will vary greatly, and the throughput will suffer severely. Although RED is widely adopted as the queue management scheme in the network, it has two major problems – unnecessary packet loss and queue size oscillation. Both will lead to the poor network performance, e.g. low link utilization and large delay. In addition, the rapid progress of wireless communications also makes TCP-Reno and RED unsuitable for current communication networks because they decrease the throughput unnecessarily and degrade the network performance in the presence of wireless link errors. To overcome the weakness of these algorithms, some new AQM and

[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.

TCP techniques were developed in recent years. However, they cannot solve all of the performance problems mentioned above. Considering that the bandwidth available to service people cannot be enlarged without a limit and that the service demands are increasing at an exponential rate, the traffic and congestion control mechanism will still be a key factor in the Internet service quality, and hence deserves more further studies.

1.2 Related Work

Originally, Internet users regularly encountered congestion in mild forms. However, severe congestion episodes also happened, and gateway congestion remained to be an obstacle for Internet applications. During several periods of 1986 and 1987, the Internet experienced the "congestion collapse" condition predicted by Nagle [Nagl84]. A large number of widely dispersed Internet sites experienced simultaneous slowdown or cessation of networking services for prolonged periods, which made the need for Internet congestion control originally apparent.

Generally, the major congestion control algorithms which have been proposed so far regarding the TCP/IP network fall into three categories according to where the algorithm functions and its mechanism. The first one is TCP-based congestion control (source control or end-to-end control), e.g. [MaSe97], in which the TCP source controls its window size according to the information it gets from the destination. The second one is AQM (router control), which regulates the queue size in the router to an acceptable level. Source control and router control usually co-exist in network traffic control to achieve high QoS (Quality of Service) requirements. The third kind of traffic and congestion control algorithms is router-feedback-based. These algorithms require the router put the information of either the router buffer status or the recommended source transmission rate into the packet header. The source then uses this information to adjust its window size to prevent congestion from happening in the router.

1.2.1 TCP-based Congestion Control

In 1987, Van Jacobson and Mike Karels developed a collection of several algorithms called Slow-start [Jaco88], which was later widely adopted as the TCP congestion control policy. Successful Internet experiences with the Slow-start algorithms (*Slow Start, Congestion*

Avoidance and *Fast Retransmit*) led them to be mandatory for TCP – TCP-Tahoe. Before Tahoe, TCP implementations applied a Go-Back-N model [Sast75], which used cumulative positive acknowledgement (ACK) and required a retransmit timer expiration to re-send lost packets. The Tahoe implementation [Jaco88] followed this model and added the Slow-start algorithms in the purpose of controlling the sender's window size according to the congestion level in the network.

The Reno TCP implementation [Stev94, Stev97, AlPa99] retained the Slow-start algorithms incorporated in Tahoe, and added a new feature in the Fast Retransmit operation – *Fast Recovery* [Jaco90]. With the Fast Recovery algorithm, the sender halves its congestion window size instead of setting it to one after Fast Retransmit. Instead of slow-starting in Tahoe, Reno uses additional incoming duplicate ACKs to clock subsequent outgoing packets. Reno's Fast Recovery algorithm can improve the link utilization in the case that only one packet is dropped in a certain window. However, it suffers from performance problems when more than one packets are dropped from one window. In this case, the sender will have to wait for a retransmit timer expiration before it can send packets again after the first fast retransmission.

To solve the performance problems of Reno in the case of multiple-packet loss from one window, a variation of Reno was proposed - New-Reno TCP [Hoe96, FlHa99], which made a small change of Reno's Fast Recovery algorithm. New-Reno does not take TCP out of Fast Recovery state upon receiving partial ACKs as Reno does. Instead, it assumes that the packets immediately following the acknowledged one have been lost. Hence, it retransmits packets without waiting for the occurrence of retransmission timeout. This increases the sender's throughput significantly when multiple-packet loss happens in one window during the data transmission.

In addition to New-Reno, there are a few other Reno extension proposed in recent years, like Forward Acknowledgement (FACK) TCP [MaMa96a], Selected Acknowledgement (SACK) TCP [MaMa96b], and Vegas TCP [BrOM94]. Researchers have done a lot of work on the TCP model and flow analysis, as well as the comparison of the algorithms mentioned above [FaFl96, BaPa97, MaSe97, Morr97].

While we appreciate the performance improvement in network congestion control achieved with various TCP algorithms, they all have the following two problems in common:

(1) they cannot improve the queue length stability in the network gateways; and (2) TCP sources cannot maintain a relatively stable transmission rate.

1.2.2 AQM

A properly designed queue management method for the Internet gateway is very important to the network performance. Good queue management techniques can utilize the link capacity more efficiently and make the network traffic stable. Otherwise, the buffer of the router can be easily overloaded or in idle state. In the former case, a large number of packets are lost and require retransmission, which will reduce the source transmission rate and may result in the system to be in the latter case. Serious packet losses can even cause network collapse and shut down the whole network.

The simplest and most commonly used algorithm in the Internet gateway is Drop Tail (DT), which drops any incoming packets from the tail of the buffer if it has been full. This algorithm has the advantage of simplicity, suitability to heterogeneity and the nature of decentralization, but it has some serious disadvantages. It lacks fairness and cannot protect the responsive sources from against non-responsive sources, which do not reduce their data transmission rate to respond to the congestion signals sent from the gateway router. Furthermore, its operation mechanism makes it difficult to balance between the link utilization and the transmission delay because it can easily trigger the global synchronization, which may cause all the sources to increase transmission rate and result in congestion in one time and to decrease their rate and cause many links underutilized in the next time. Similar to Drop Tail, Drop Head (DH) algorithm has the same advantages and disadvantages.

As the effort to solve the problems of the gateway congestion control algorithm of DT/DH plus FIFO (First in First out), Active Queue Management (AQM) has been proposed to manage the queue size in the gateway router dynamically. Instead of waiting until buffer overflow to drop the packets in DT/DH, AQM algorithms drop incoming packets randomly according to the output of some probability function. The key idea of AQM algorithms is to convey congestion notification early to the network traffic sources in order for the sources to reduce their transmission rate before the router buffer gets overflown and has to drop all the incoming packets. Therefore, compared with static queue management methods like DT and

DH, AQM algorithms should be able to achieve better queue size control performance, higher link utilization and shorter traffic delay.

Random Early Detection (RED) (Ref: Appendix 2) is the well-known and most widely adopted AQM algorithm. Since it was proposed by S. Floyd and V. Jacobson in 1993 [FlJa93], it has attracted a lot of researchers to work on its theoretical analysis and performance evaluation [FiBo00, TiMa01, HoMi01a, KuLa01, LoPa02]. In RED, there are some important parameters need to be set properly to make it perform well. While RED outperforms DT or DH significantly, it has a few obvious problems. First, early congestion notification is built on the sacrifice of unnecessary packet loss. Second, RED is unable to control the queue size very well because the queue sizes under the control of RED may oscillate dramatically, which is one phenomenon of network instability.

Quite a few algorithms extended from RED have been proposed in the effort of improving the RED algorithm. These algorithms differentiate from one to another, but they all share the basic idea of RED – starting to discard packets before congestion really happens. For example, DRED (Dynamic RED) [AuOu01] comes from the classical control theory. Its goal is to maintain the queue size close to a threshold value, which is independent of the traffic load on the network, by using a closed-loop feedback controller to adapt the packet-dropping probability as a function of the average distance of the queue from the threshold value. Like RED, SRED (Stabilized RED) [OtLa99] preemptively discards packets with a load-dependent probability when the buffer is approaching congested. In addition, SRED can stabilize the buffer occupation at a level independent of the number of active connections through the information collection of active connections or flows and the identification of flows that are misbehaving (taking more than their fair share of bandwidth). FRED (Flow RED) [LiMo97] considers the fairness issue for different types of traffic and imposes on each flow a loss rate according to how much the flow occupies the buffer, instead of applying the same dropping rate to all the flows. If a flow continually occupies a large part of the buffer space, then it will be detected and limited to a small part of buffer occupation. ARED (Adaptive RED) [FeKa99] configures its parameters according to the traffic load. If the average queue size is between the minimum buffer threshold and maximum buffer threshold, its maximum packet dropping probability will be multiplicatively scaled up or down by two different factors based on the traffic load status. The purpose of ARED is to set the

parameters in response to the traffic load change automatically. Unlike RED, whose queue length give little information about the number of competing connections sharing a certain link, BLUE [SaFe99] was designed to use the events of packet loss and link being idle to protect TCP flows against non-TCP (non-responsive) flows. After the queue size has exceeded a certain value for a given period of time (freeze time), the drop probability will increase by a constant factor. On the other hand, if the link remains idle for the freeze time, then the drop probability will decrease by another constant factor. In the PI-RED (Proportional and Integral Controller in RED Router) [HoMi01b] (named for short in this thesis), the packet drop probability is not randomly determined by a probability function like the case of RED, but is calculated by a proportional and integral (PI) controller. The PI controller controls the averaging buffer queue size at the reference level. If the difference between the averaging queue size and the reference queue size exists, then the PI controller will verify its output to change the packet drop probability.

All the variants of RED mentioned above can improve the performance of RED in various ways, but the queue size oscillation problem can still be seen. In addition, These AQM congestion control algorithms, including RED, are all intended to stabilize the queue size in the gateway. They notify the senders of the possible congestion status by dropping packets. Because wireless link errors can also result in packets being dropped, the senders then have no way to judge correctly if a packet loss is caused by the network congestion or by the wireless link errors (to be described in Section 1.2.4). This will inevitably affect the network performance negatively when it involves the wireless transmission fully or partially.

1.2.3 Router-Feedback-based Approaches

Both TCP congestion control algorithms (e.g. Reno TCP) and AQM algorithms (e.g. RED) build their congestion control mechanism on the information obtained locally or from the other end implicitly. For TCP Reno sources, the RED router acts like a black box. The TCP source can only make its judgment about the network work status by the implicit information it receives from the router (dropped packets). On the other hand, TCP Reno sources are also the black boxes to the RED router. The router cannot help the TCP sources explicitly to set their congestion control schemes more efficiently. For this reason, the RED and Reno

combination can never achieve the optimal performance for link utilization and gateway router queue size.

Many researchers have noticed this problem and proposed new schemes that explicitly convey the information of the router buffer occupancy in the TCP header to the TCP senders to let the senders make precise regulation of their data transmission rate. This group of congestion methods is named Router-Feedback-based approaches in this thesis because they all fall into the concept that the information about the gateway router is fed back to the source. The conveyed information is either a single digital bit which indicates if the router is congested or not or the exact bandwidth available in the router.

Sally Floyd proposed Explicit Congestion Notification (ECN) (Ref: Appendix 3) [Floy94] to solve the unnecessary packet loss problem of RED. An ECN router marks the packets to inform the TCP sources to reduce their transmission rate, instead of dropping the packets, if it is congested. This solves the unnecessary packet loss problem of RED. However, the buffer queue length in the router with ECN is as oscillated as RED, because ECN only tells the sender if the router is congested or not. It does not give the information of the exact bandwidth available in the router. The performance analysis of ECN can be found in a lot of research papers, e.g. [KaKe00].

To utilize the link bandwidth optimally, some new ideas that originated from the ATM (Asynchronous Transfer Mode) network traffic control have been proposed to control TCP congestion in recent years. For example, classical control theory and *Smith's principle* were proposed as key tools for designing a congestion control law for high-speed data networks [Masc99]. However, this controller had to solve the fairness problem by "per-flow queuing with round-bin scheduling", which is costly to implement. A proportional feedback-based rate controller was designed to control the traffic flow for ATM networks [ZhYa97] and TCP networks [ZhYa00], with the stability criteria presented and proved with Lyapunov Second Stability Criterion in modern control theory. The limitation of the proportional rate controller is the steady-state error problem [CoZh03].

In addition to those control-theory-based traffic control algorithms, some simulation-based algorithms were also proposed recently [GeWe00, AwOu02, KaKa00]. They are similar to each other in that all of them compute the transmission rate according to the buffer occupancy of the bottleneck router and then control the source transmission with this

computed rate. The stability with those controllers was only validated with simulation results and lacks any theoretical performance proof.

1.2.4 TCP over Wireless Links

With the development of the computer communication networks and the marvelous market demands, the wireless communication has been merging into the traditional wired communication system and plays a very important role in today's network. The merging of wired and wireless communications has brought a new challenge to the network performance. Originally, all the network congestion control algorithms were designed only to serve the wired communication system, in which it is assumed that congestion in the network is the primary cause for packet losses. TCP performs well over such networks by adapting to end-to-end delays and packet losses caused by congestion, though there is still some performance problems. However, in a typical wireless communication system, it is common that packets are lost because of the link error problem. When packets are lost for reasons other than congestion, the TCP sources unnecessarily reduce their end-to-end throughput. For this reason, the network performances suffer from the link-error-caused packet losses.

This performance degradation of TCP over wireless has drawn much attention from researchers in the network communication field since the middle of 1990s. Several schemes have been proposed to alleviate the effects of non-congestion-related packet losses on TCP performance over networks that contain wireless links.

Snoop [BaSe95] is one of these schemes that address the performance problem of TCP over wireless links. The Snoop protocol introduces a new module, which is called the *snoop agent*, at the base station, so that the retransmission due to packet loss is only conducted over the wireless link and the wired links do not pass the duplicate ACKs to the TCP source. Hence, the TCP source does not reduce window size unnecessarily.

Another well-known scheme is TCP Westwood [CaGe00, CaGe01, GeSa01], which uses the bandwidth estimation to determine the congestion window size value at the TCP sources when the fast retransmit function is activated. The throughput of the TCP senders can be improved with TCP Westwood. TCP Westwood has the advantage that it only needs the sender-side modification of current congestion control schemes and is totally transparent to the router, but is cannot help to stabilize the queue size in the router.

The algorithms aimed at improving the TCP performance for network containing wireless links are classified into three groups [BaPa97] – end-to-end proposals, split-connection proposals and link-layer proposals. This group of end-to-end algorithms allows the TCP sender to tell the difference between congestion-caused packet loss and wireless-link-related loss. The split-connection group completely hides the wireless link from the source by terminating the TCP connection at the base station and use a separate reliable connection between the station and the destination. The third group of algorithms, link-layer group, hides the link-error-caused packet losses from the TCP source by using local retransmissions over wireless links. The local retransmission is conducted with the techniques that can increase the performance significantly. Paper [BaPa97] also provides performance analysis and comparison of TCP over wireless links for these three groups of algorithms. The link layer proposals have the problems of the competing retransmissions between the two layers and the negative effect of link layer protocol on the TCP fast retransmission. The others only concern the TCP performance, and cannot benefit the gateway performance at the same time.

1.3 Thesis Motivation

Since V. Jacobson proposed Tahoe in 1988, the TCP congestion control algorithms have been upgraded several times, including Reno, New-Reno, SACK, FACK, Vegas, and Westwood. These new TCP mechanisms have improved the TCP throughput and congestion control performance to a higher level, but as documented before they can do nothing to control the queue size stability in the routers, which appear as the black boxes to TCP senders. On the other hand, AQM algorithms can control the router buffer occupancy rate within an acceptable range to prevent network from congestion in the gateways. However, they are unable to keep a relatively stable queue size in the router, which is beneficial to the ever-increasing real-time streams on the network, e.g. Internet audio players, IP-telephony, and video conferencing. Furthermore, the current TCP plus AQM network operation mode (e.g. TCP Reno + RED) suffers from the performance degradation when the network involves communications over wireless channels.

Based on the above, it is desirable to have a traffic and congestion control scheme that satisfies the following three requirements: (1) can stabilize the queue size in the router; (2)

can avoid the unnecessary packet loss; and (3) can improve the TCP performance over wireless links. The motivation of this thesis is to propose such a scheme with the easy method of implementation. Besides, we would like to prove our proposal in modern control theory, as we have not seen any similar theoretical proof from others so far.

1.4 Objectives

In general, we are interested in designing a network traffic and congestion control scheme that can meet the three conditions mentioned in the last section. However, the research in this thesis will focus on the following aspects:

1. Propose a new traffic and congestion control method.
2. Theoretically analyze and validate the new traffic and congestion control method.
3. Compare the performance of the new traffic and congestion control method with some other common methods under various network scenarios.

1.5 Methodologies and Approaches

To achieve the goals mentioned in the last section, a great deal of simulation work needs to be done to study the network behavior in different scenarios. All the simulations are based on the simulation tool – OPNET [OPNE00]. OPNET is an engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis. It is a mature commercial tool and has a complete user-oriented procedure for installation and simulation development, which makes it the best choice for the simulation work in this thesis.

The major simulation models we developed in OPNET can be divided into three different layers: the network configuration models, the node models and the process models. In addition, we also created some other models, including the packet format, the probe model, and the simulation sequence, to obtain various statistics and evaluate performance.

We studied the congestion control algorithms in two different bottleneck network configuration models: the wired network model and the mixed (wired/wireless) network model to see how they perform in different communication environments. We create bottleneck router to study congestion and evaluate the performance of congestion control in

terms of queue size stability, packet drop rate and throughput in seven different scenarios (four wired network scenarios and three mixed network scenarios) by simulations.

Control theory is not new in the network area. Many researchers have applied the proportional/proportional and integral controller to the traffic/congestion control in various ways. We would like to study the congestion control problem in a network running TCP traffic in continuous-time and model the network traffic as a fluid flow as done in [ZhYa97, ZhYa00, ZhCo03]. We want to control the network congestion by regulating the transmission rate in the sources according to the queuing information fed back from the intermediate nodes.

In designing the new traffic and control scheme – the PIR controller, we first investigate the Proportional controller, which serves as the base of our PIR controller. Then we introduce the integral part to the original proportional controller in order to eliminate the steady state error problem of the proportional controller (We do not want to design a more complicated proportional-based controller such as the PID (Proportional, Integral and Differential) controller, as this is very difficult for tuning in the reality). We prove with Lyapunov Second Stability Criterion in modern control theory that the PIR controller is feasible for the network traffic and congestion control. Finally, we evaluate and compare the performance of our PIR controller with other congestion control schemes by simulations.

1.6 Thesis Contributions

The following are the contributions of this thesis:

1. Proposal of a PIR controller called for network traffic and congestion control.
2. Theoretical stability proof, implementation method, and simulation-based performance evaluation of the PIR controller.
3. Performance analysis and comparison of RED, ECN, PI-RED and PIR controller in wired and wireless communication environments.
4. Development of the OPNET simulation models.
5. Implementation of various congestion control algorithms, especially the Proportional and PIR controller, for analysis and comparison.

1.7 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 describes the network operation, models and assumptions for the wired network model, the mixed (wired/wireless) network model, the proportional control mechanism, the short-lived TCP flow, and the long-lived TCP flow. In Chapter 3, we prove the controller stability theoretically and provide two implementation methods of the PIR controller. Chapter 4 presents the performance evaluation and comparison for RED, ECN, PI-RED and our PIR controller in the wired network environment under four different simulation scenarios. The performance evaluation and comparison for RED, ECN, PI-RED and the PIR controller in the mixed network environment under three other different simulation scenarios are presented in Chapter 5. In Chapter 6, a design guideline for the PIR controller is provided. Chapter 7 concludes the thesis and provides some suggestions for the potential future work.

1.8 Publications

The following are publications resulted from our research work:

1. Jun Cong, Hongyi Zhang and Oliver Yang, "A Queuing Performance Comparison of Extended Proportional Controller in TCP Traffic Control", *Proceedings of CCECE 2003*, vol.2, pp. 973-976, Montreal, Apr 2003.
2. Hongyi Zhang, Jun Cong and Oliver Yang, "Rate Control over RED with Data Loss and Varying Delays", *Proceedings of Globecom 2003*, San Francisco, Dec 2003, Session NG11-4.
3. Jun Cong, Oliver Yang and Hongyi Zhang, "Enhancing the TCP Traffic Control Performance with A Proportional and Integral Rate Controller", to be presented at *ICC 2004*, Paris, France, Jun 2004.

Chapter 2
Network Operation, Models and Assumptions

There are different kinds of communication network environments. Each of them has its own operation and characteristics. In this chapter, we will introduce the network operation, the network models, the background theory, the traffic flow models, the OPNET models, and the assumptions that are used in this thesis.

2.1 Network Operation

We envision end to end communication using TCP through a number of intermediate routers, each implemented with the traffic congestion control algorithm. A TCP source generates and sends a packet, which contains the source address and 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 TCP source and sends the packet to the destination or the next intermediate router if it is not next to the TCP destination. As a result, the routers will finally forward the packet to the corresponding TCP destination according to the destination address they get from the IP header of the packet if the packet is not discarded. The destination, upon receipt of the packet, generates the ACK and sends it back to the source.

There are two different types of network we consider in this thesis: the Wired Network and the Mixed (Wired/Wireless) Network, on which we have created many scenarios later on to do our study and investigations.

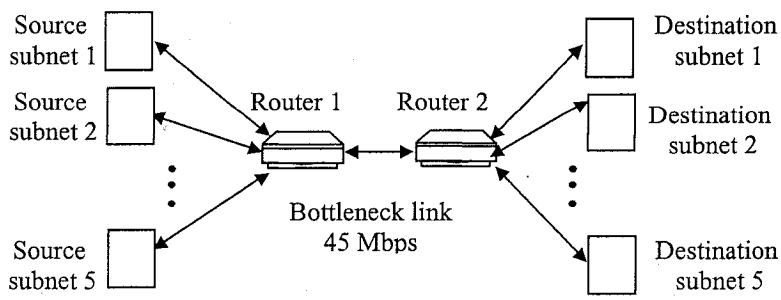


Figure 2-1 Wired Network

2.1.1 Wired Network

Figure 2-1 shows a simple example of a wired network with 5 source subnets and 5 destination subnets. Each source/destination subnet has 20 TCP sources/destinations (totally 100 sources and 100 corresponding destinations). The sources and the destinations are connected with a T3 (45 Mbps) link. The TCP sources in one subnet have the same propagation delay. The propagation delay for the TCP sources in one subnet is different from that of each other subnet. The RTT is composed of the propagation delay and the queuing delay, which is varied. A bottleneck congestion situation can occur in Router 1 if the packet arrival rate exceeds the packet service capacity. Note also that there are different delays for different TCP subnets.

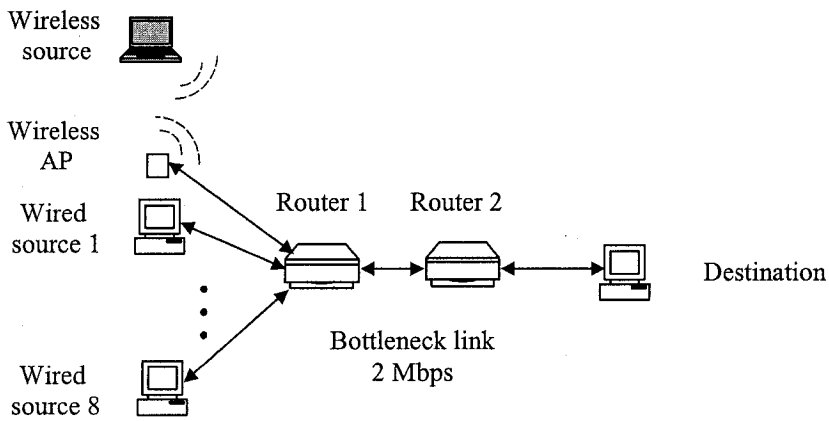


Figure 2-2 Mixed (Wired/Wireless) Network

2.1.2 Mixed (Wired/Wireless) Network

Figure 2-2 shows a simple example of a mixed (wired/wireless) network with totally 9 TCP source nodes (One of them is a wireless node, connected to Router 1 through the AP (access point)) and 1 wired destination node. All the sources communicate with the same destination. The sources and the destination are connected with a 2 Mbps bottleneck link. All the connections have the same propagation delay.

2.2 Traffic Control Mechanism

As we have mentioned in Section 1.5, we created a bottleneck router to study the queuing performance for various congestion control schemes, RED, ECN, PI-RED and PIR controller, which we proposed in this thesis. The bottleneck router for these schemes can be described by the following analytical model, which we present below to introduce the model, symbols and notations to be used later.

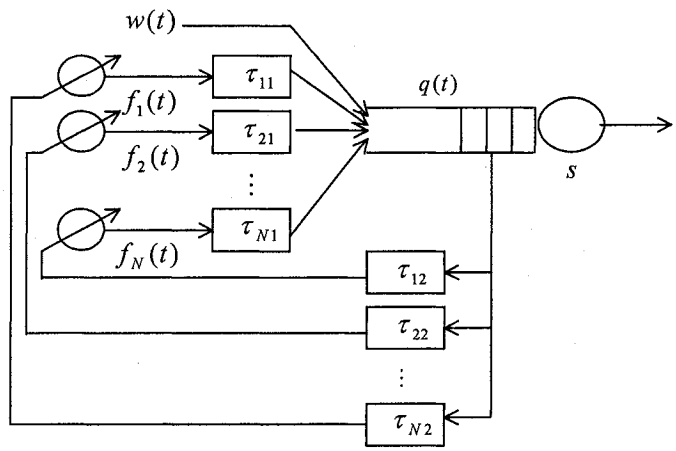


Figure 2-3 Analytical Model of One Bottleneck Network

Figure 2-3 above depicts a single-bottleneck network with N TCP connections and a non-controlled connection. These connections go through a bottleneck router, as shown by the server and its buffer. Divide the round-trip delay for a TCP connection into two parts: (1) τ_{i1} is the path delay from the ith source to the bottleneck router, and (2) τ_{i2} is the delay from the bottleneck router back to the ith source through the destination. Let $f_i(t)$, $w(t)$, and s be the transmission rate of TCP source i, the rate of non-TCP-controlled source packets entering the router, and the bottleneck router service rate, respectively. Then, the instantaneous queue size $q(t)$ in the bottleneck router can be described as

$$q(t) = \int_0^t [\sum_{i=1}^{N} a_i(v-\tau_{i1}) f_i(v-\tau_{i1}) + w(v) - s]dv \quad , \qquad (2\text{-}1)$$

where $a_i(t)=1$ if the source is active. Otherwise, $a_i(t)=0$. The term $a_i(v-\tau_{i1})f_i(v-\tau_{i1})$ represents the rate at which Source i sends packets into the bottleneck router. The term $s-w(v)$ is the bandwidth available in the bottleneck router to serve the TCP data traffic.

Our PIR controller is based on a continuous-time feedback proportional controller, which we introduce as follows.

Let q_{SP} be the pre-set reference queue size (desired queue size). Then the dynamics of the instantaneous queue size error $e(t)=q(t)-q_{SP}$ can be shown to be

$$\dot{e}(t)=\sum_{i=1}^{N}a_i(t-\tau_{i1})\mu_i(t-\tau_i)+w(t)-s \quad , \tag{2-2}$$

where $\dot{e}(t)$ is the first order differential of $e(t)$, $\mu_i(t)=f_i(t+\tau_{i2})$ and τ_i is the round-trip delay for Source i.

Based on Equation (2-2) and let $\mu_i^{\max}$ be the peak transmission rate recommended for TCP source i, then a proportional controller can be defined as

$$\begin{aligned}\mu_i(t) &= \mu_i^{\max}-k_i e(t)\\ &= \mu_i^{\max}-k_i[q(t)-q_{SP}] \quad , \qquad i=1,2,\cdots,N \quad ,\end{aligned} \tag{2-3}$$

where k_i is the proportional gain for Source i. Therefore, with the regulation by the continuous-time proportional controller, the data transmission rate at TCP source i is bounded as

$$f_i(t)=\mu_i^{\max}-k_i[q(t-\tau_{i2})-q_{SP}] \quad , \quad i=1,2,\cdots,N \quad . \tag{2-4}$$

As for the traffic control mechanisms of the other three congestion controllers to be compared with our PIR controller in this thesis, the detailed descriptions can be found in Appendix A2, A3 and A4 for RED, ECN and PI-RED, respectively.

2.3 TCP Traffic Flows

There are two types of traffic flows studied in this thesis: the short-lived TCP flow and the long-lived TCP flow. Both types of the TCP flow sources have a transmission rate limited by the traffic control algorithm. For RED, ECN, PI-RED and one of the PIR controller

implementation we proposed (PIR-B, see Chapter 3 for details), the transmission rate of a TCP source is controlled by a TCP congestion control and avoidance algorithm - TCP Reno. For the other PIR controller implementation (PIR-A, see Chapter 3 for details), the traffic rate of TCP sources is determined by the advertised rate which is computed in the bottleneck router and is conveyed back to TCP sources in ACKs.

2.3.1 Short-lived TCP Flow Model

The short-lived TCP flow is characterized by the uncertainty of the time of its entering/reentering the network and the time length of its being active. HTTP (Hypertext Transfer Protocol) source is a typical example of the short-lived source, which enters the network after an undetermined think time, sends a file of an undetermined length, and then waits for another think time period.

Much study has been done by researchers on the traffic pattern of the shorted-lived TCP flows, e.g. [CrBe97], and various probability functions were proposed to describe the characteristics of the think time and file length distributions, such as the self-similarity model [LeTa94] and the wide area traffic model [PaFl95]. For our work, we have adopted the model from [OtLa99]. The think time characteristic is depicted by the following probability distribution

$$P\{T > t\} = p_h\left[1+(\frac{t}{\pi_h})^{\eta_h}\right]^{-1} + p_l\left[1+(\frac{t}{\pi_l})^{\eta_l}\right]^{-1}, \qquad (2\text{-}5)$$

where p_h, p_l, π_h, π_l, η_h and η_l are constant parameters, and $p_h + p_l = 1$. The file length (in bytes) probability distribution is

$$P\{F > f\} = \left[1+(\frac{f}{\pi})^{\alpha}\right]^{-1}, \qquad (2\text{-}6)$$

where π is the median and α is a positive constant.

The distribution of Equation (2-6) has a finite mean but an indefinite variance. The distribution of Equation (2-5) is a mixture of two distributions of Equation (2-6). It has been shown that the traffic pattern of short-lived TCP flows following these two distributions is realistic [OtLa99].

The transmission rate of the short-lived TCP source is limited by the real-time bandwidth available on the network. Figure 2-4 shows the transmission pattern for a typical short-lived TCP flow, which we have verified to be commensurate to real traffic from the Internet.

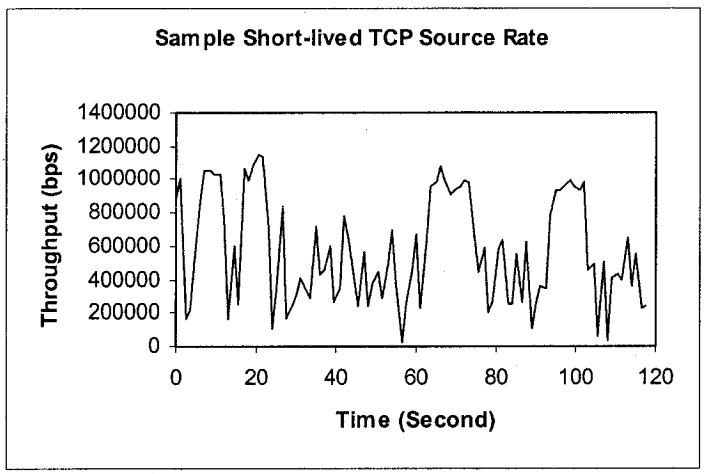


Figure 2-4 Transmission Rate of Sample Short-lived TCP Flow

2.3.2 Long-lived TCP Flow Model

Unlike the short-lived TCP source, the long-lived TCP source will always have data to send and never stop sending as long as it is allowed to do so by the congestion window size. A good example of the long-lived TCP source is the greedy FTP (File Transfer Protocol) sender, which transfers an indefinitely long file. Like the short-lived TCP flow, the rate of the long-lived TCP flow is limited by the real-time bandwidth available for it.

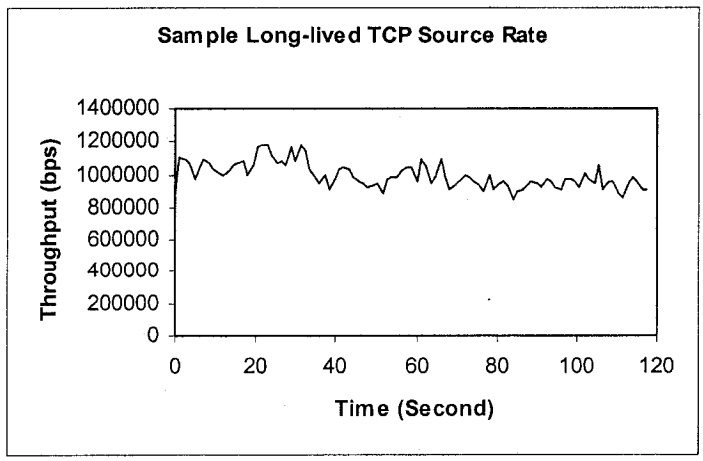


Figure 2-5 Transmission Rate of Sample Long-lived TCP Flow

Figure 2-5 shows the transmission pattern for a typical long-lived TCP flow under the congestion control of 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 the hierarchical and object based modeling.

Models built with OPNET are hierarchical. At the lowest level, the process domain is structured as a Finite State Machine (FSM). The FSM can be structured with the help of a graphical editor that allows the user to specify the relationship between the states. The states can then be programmed with a C like language called Proto-C. The second level is the node level. Processes that were specified in the process domain, different source and destination modules offered by OPNET, as well as data generators and queues can then be grouped into nodes in the node domain. Nodes can then be connected to build up different network architects in the network domain.

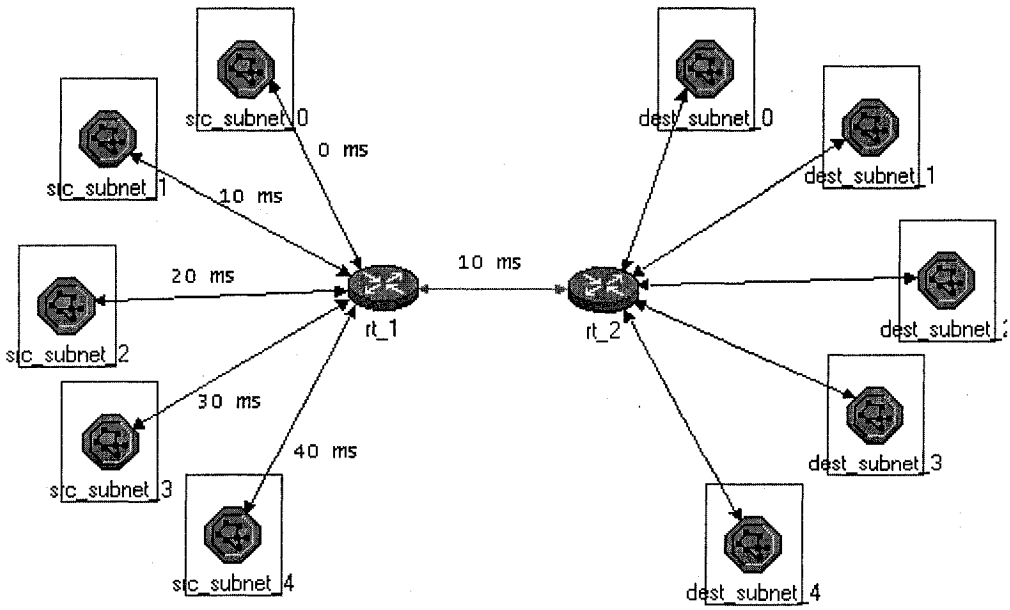


Figure 2-6 OPNET Network Model – Wired Network

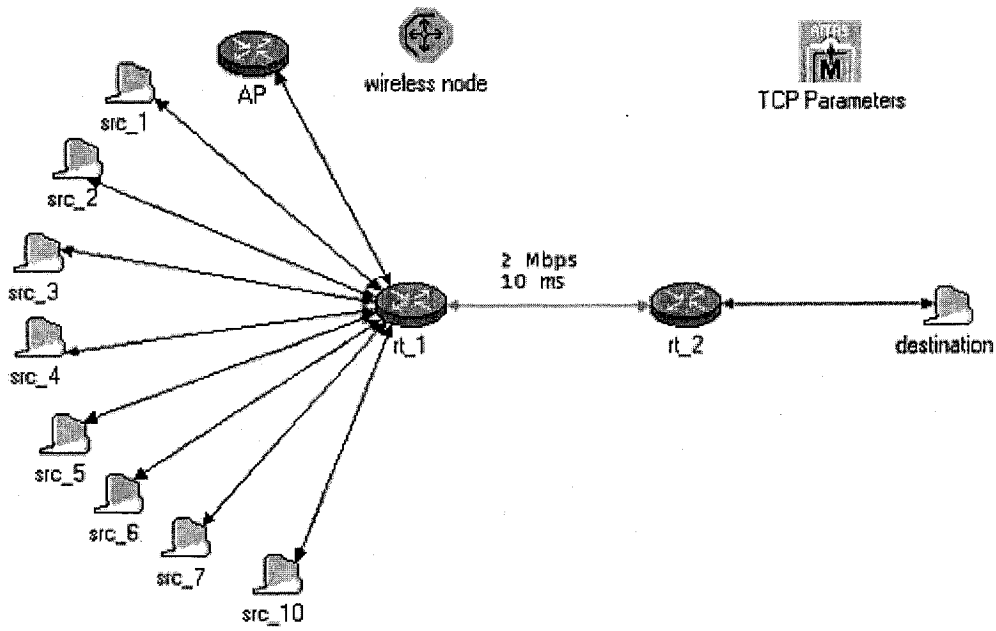


Figure 2-7 OPNET Network Model – Mixed Network

Figures 2-6 and 2-7 show the OPNET network models that are built for the wired network configuration and the mixed (wired/wireless) network configuration used in the thesis, respectively. The appearance of OPNET network models is intuitive. We can find they are quite similar to that of the wired network model and the mixed network model shown in Figures 2-1 and 2-2. The node "TCP Parameters" in both figures stores some preset simulation parameters, e.g. the upper bound window size for TCP sources, which is set to be 10000. In Figure 2-6, each source subnet has 20 TCP source nodes and one multiplexer node, and each destination subnet has 20 TCP destination nodes and one multiplexer node. In the mixed network model shown in Figure 2-6, we use wireless radio transmission/receiver pipelines in OPNET to setup the connection between the wireless node and the AP. By adjusting the physical distance between the wireless node and the AP and the radio transmission power, we obtained different packet error rates for the wireless channel.

Figure 2-8 shows the OPNET node model for the TCP source. We can see that there are three different objects in the TCP node model. The TCP source generates packets according to some particular TCP mechanism and sends them to the transmitter, which is an OPNET

built-in object. The transmitter forwards the packets outside to the network link. The receiver, also an OPNET built-in object, receives ACKs, which are transmitted to the node through the network link, and forwards them to the TCP source. The TCP source then processes the ACKs and responds accordingly.

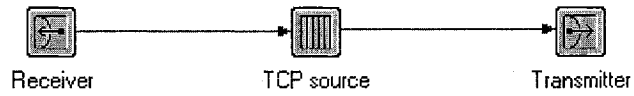


Figure 2-8 OPNET Node Model – TCP source

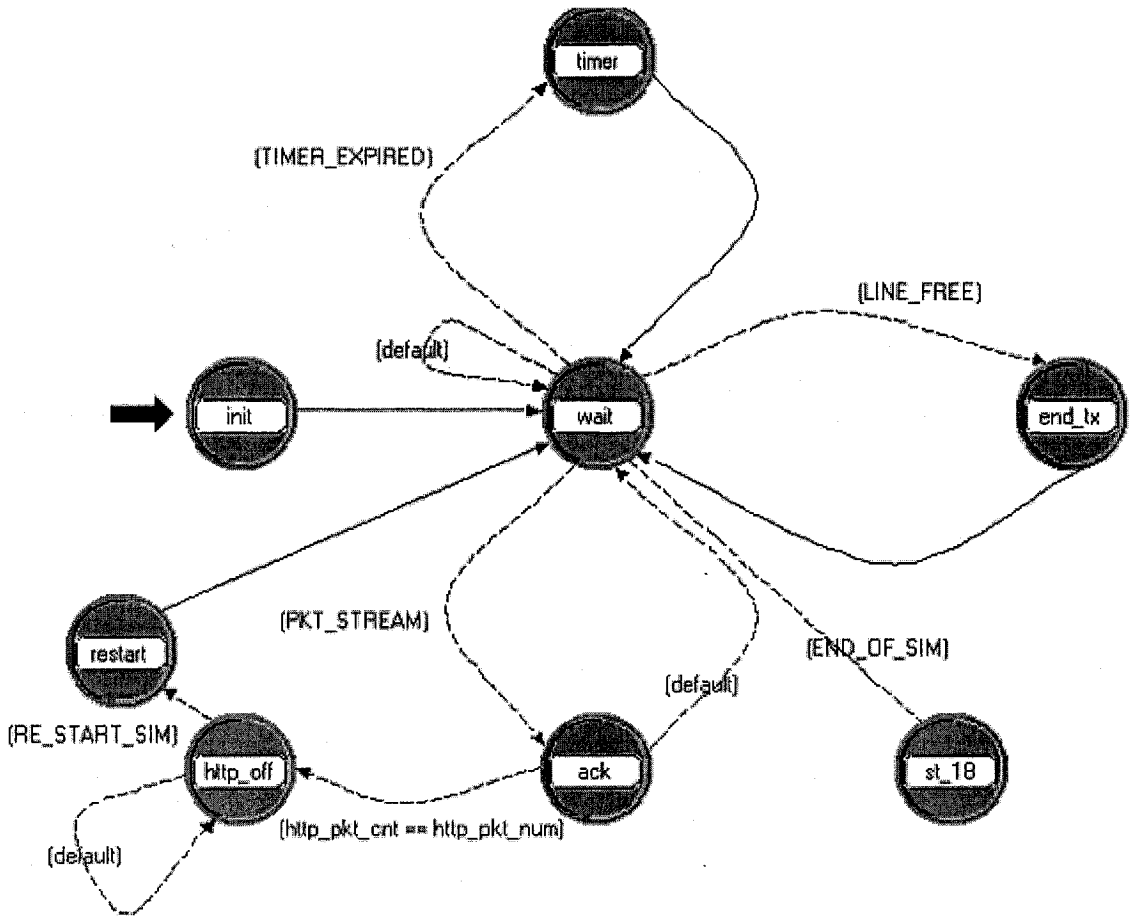


Figure 2-9 OPNET Process Model – TCP source

Other OPNET node models, such as the TCP destination and the router share the same mechanism.

Figure 2-9 shows the OPNET process model of the TCP source. The process model includes all the possible states in which the TCP source can be. For example, when the simulation begins, the system is in the *init* state first, and the simulation is initialized. Then the system goes into the *wait* state. If the TCP source node receives an ACK, then it enters the *ack* state to respond accordingly.

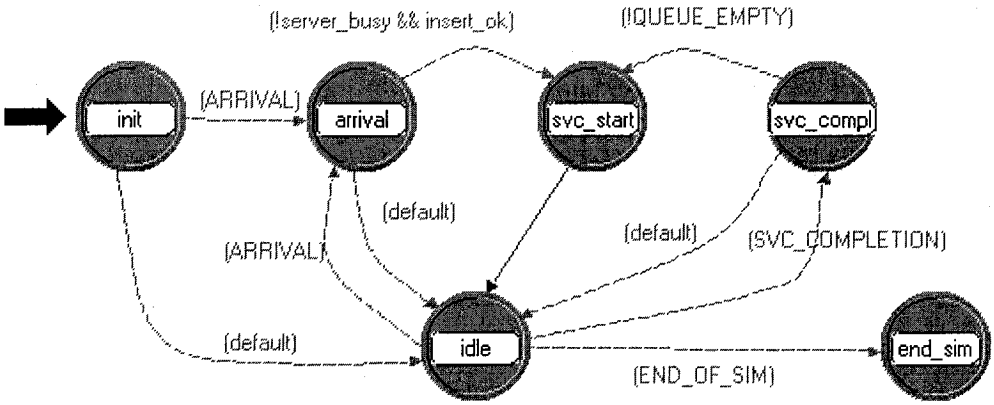


Figure 2-10 OPNET Process Model – Router Packet

Figure 2-10 shows the OPNET process model of the router packet. Intuitively, we can see how the router works for incoming packets from its process model. After receiving a packet, the router enters the *arrival* state, and then goes into the *svc_start* state if the server is not busy and the packet is successfully inserted. After the server finishes processing the packet and no more packets are waiting for service, it enters the *idle* state from the *svc_compl* state.

In addition to the TCP source process model, the TCP destination process model and the router process model, there are still some other OPNET process models we used for the studies in this thesis. They are: the source subnet multiplexer process model, the source subnet demultiplexer process model, the router ACK process model, the the TCP destination

process model, the destination subnet multiplexer process model, the destination subnet demultiplexer process model, and the global parameter process model.

The OPNET models we created also include the link model, the packet format model, the probe model and the simulation sequence. They are necessary parts of the OPNET network models. All these models together become the whole set of simulation models we used to do simulations in the thesis.

2.5 Assumptions

The following assumptions are used throughout this thesis:

1. The 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 the performance comparison;
2. The buffer size of Router 2 (Node rt_2 in Figure 2-6 and 2-7) is big enough so that no packet loss will occur there, while the buffer size of Router 1 (Node rt_1 in Figure 2-6 and 2-7) is made finite. This assumption allows us to focus on the performance comparison in a one-bottleneck-router network;
3. All the packets have the same size;
4. The TCP destination's receiver window size is large enough that it does not affect the window size of the TCP source. This would allow us to focus on the queuing performance evaluation of the congestion control algorithms residing in the router only.

For the wired network, we have the following in addition:

5. Each source and its counterpart destination are one to one mapping and do not have cross traffic. This would allow us to simplify the simulation and evaluation of the queuing performance in the bottleneck router;
6. Each subnet has the same ratio of ftp sources to http sources for the simplicity of running simulation;

and for the mixed network, we have in addition

7. No link layer approaches are applied to improve the TCP performance over wireless links because the throughput performance comparison cannot be obtained directly regarding the various higher layer algorithms, otherwise.

Chapter 3
Proportional and Integral Rate Controller

In this chapter, we will propose a new network traffic congestion control scheme – the Proportional and Integral Rate (PIR) controller. First, we will introduce the network operation mechanism specific to the PIR controller. Then, a theoretical proof of the PIR controller's stability in modern control theory will be presented. Finally, we will provide two implementation methods for our PIR controller.

3.1 Control Mechanism

With the proportional controller introduced in Section 2.2, the queue size in the bottleneck router is monitored and regulated continuously in a continuous-time fluid regulation way, a high piecewise stabilization of the queue size can be achieved and maintained. The limitation of the proportional controller is that it cannot solve the problem of steady state error. Like all other proportional controllers, when the traffic load on the link varies greatly, the proportional controller will have to keep some steady state error in order to make the controller output change accordingly to obtain the network stabilization.

The departure point of our new controller from the traditional design [ZhCo03] is based on Equation (2-3), where we use a different $\mu(t)$ for our controller. That is

$$\mu(t) = \mu_{\max} - k[q(t) - q_{sp}(t)] \quad , \tag{3-1}$$

where $q_{sp}(t)$ is the varied queue length setpoint at time t. The parameters $\mu(t)$, $\mu_{\max}$ and k are the global recommended source rate, the global recommended peak source rate and the global proportional gain for all the connections, respectively.

Note that $q_{sp}(t)$ is not equal to q_{SP} (the reference queue size) necessarily. Each time when the packet received in the router is ready to be sent out, $q_{sp}(t)$ is revised as follows according to the instantaneous queue size in the router.

$$q_{sp}(t) = \begin{cases} q_{sp}(t^-) + d \cdot |q(t) - q_{SP}|, & if \quad q(t) < q_{SP} \\ q_{sp}(t^-), & if \quad q(t) = q_{SP} \\ q_{sp}(t^-) - d \cdot |q(t) - q_{SP}|, & if \quad q(t) > q_{SP} \end{cases}, \tag{3-2}$$

where d is the adaptation constant and t^- is the time when $q_{sp}(t)$ is last updated.

The purpose of Equation (3-2) is to maintain a larger queue size setpoint when the real queue size is less than the reference queue size, and a smaller queue size setpoint when the real queue size is larger. Thus the real queue size can be kept stable at the reference size level.

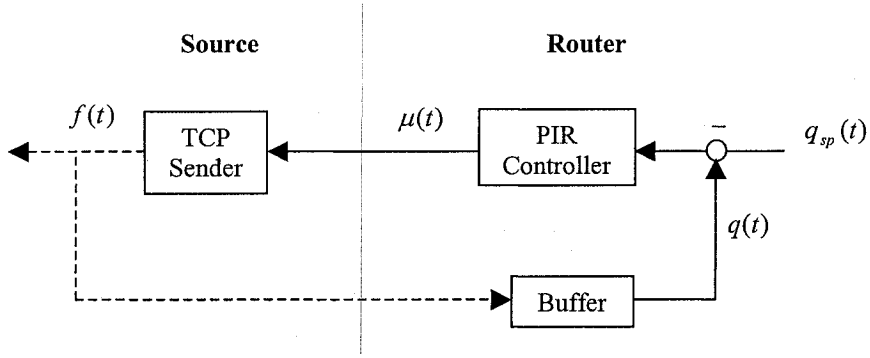


Figure 3-1 Block Diagram of PIR Controller Feedback System

Figure 3-1 is the block diagram of the PIR controller feedback system. The PIR controller uses the buffer queue size $q(t)$ and the queue size setpoint $q_{sp}(t)$ to compute the controller output $\mu(t)$, which in turn controls the rate in the source. The source then adjusts its traffic rate $f(t)$ and affects the queue size $q(t)$ in the router. Hence, a closed-loop feedback control system is formed. The solid lines in the figure indicate the paths of control information, while the dashed ones are actual flows in the physical media.

This controller can avoid congestion in the following manner. When the queue size is getting large, the sources will take steps to regulate its transmission rate, as they have received the information from former ACKs coming from the destination. Even though packets have to be dropped in some cases (due to large bursty traffic appearing on the link suddenly), they are dropped only after the queue exceeds the maximum buffer threshold, thus no packets will be dropped unnecessarily. In addition, as the queue size in the router is monitored and regulated continuously in a feedback control system, the PIR controller can

also achieve stabilization of queue size and source transmission rate, as well as high link utilization.

3.2 Stability Analysis

We can prove the stability of our PIR controller and its equilibrium conditions, based on Lyapunov Second Stability Criterion (Ref: Appendix A1) in modern control theory.

According to the control mechanism of the PIR controller, each time when the router transmits a packet, $q_{sp}(t)$ is updated before the transmission. This updating rate can be expressed by the rate at which the bottleneck router serves the TCP data traffic. Since the queue is congested, one can see that this rate can be approximated by s, the packet service rate of the bottleneck router. Thus, Equation (3-2) can be expressed as follows.

$$q_{sp}(t) = q_{SP} - \int_0^t e(v) \cdot sd \cdot dv \quad , \tag{3-3}$$

where d is the adaptation constant.

Substituting $q_{sp}(t)$ in Equation (3-1), we obtain

$$\mu(t) = \mu_{\max} - k\left[q(t) - q_{SP} + \int_0^t e(v) \cdot sd \cdot dv \right]$$

$$= \mu_{\max} - ke(t) - k\int_0^t e(v) \cdot sd \cdot dv \quad . \tag{3-4}$$

Define the variable $p(t)$ such that $\dot{p}(t) = \frac{dp(t)}{dt} = sd \cdot e(t)$. Then, Equation (3-4) can be rewritten as

$$\mu(t) = \mu_{\max} - ke(t) - k\int_0^t \dot{p}(v) \cdot dv$$

$$= \mu_{\max} - ke(t) - kp(t) + kp(0) \quad . \tag{3-5}$$

Let $m = sd$, $n_i(t) = -ka_i(t - \tau_{i1})$ and $\xi(t) = \sum_{i=1}^{N} a_i(t - \tau_{i1})\left[\mu_{\max} + kp(0)\right] + w(t) - s$.

From Equation (2-2), we can have

$$\begin{cases} \dot{e}(t) = \sum_{i=1}^{N} n_i(t)e(t-\tau_i) + \sum_{i=1}^{N} n_i(t)p(t-\tau_i) + \xi(t) \\ \dot{p}(t) = me(t) \end{cases} . \tag{3-6}$$

Equation (3-6) in matrix form can be written as

$$\begin{bmatrix} \dot{e}(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ m & 0 \end{bmatrix} \begin{bmatrix} e(t) \\ p(t) \end{bmatrix} + \sum_{i=1}^{N} \left\{ \begin{bmatrix} n_i(t) & n_i(t) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e(t-\tau_i) \\ p(t-\tau_i) \end{bmatrix} \right\} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \xi(t) . \tag{3-7}$$

The term $\xi(t)$ in Equation (3-7) can be seen as the system input, so the stability of the above system is equivalent to the following system:

$$\begin{bmatrix} \dot{e}(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ m & 0 \end{bmatrix} \begin{bmatrix} e(t) \\ p(t) \end{bmatrix} + \sum_{i=1}^{N} \left\{ \begin{bmatrix} n_i(t) & n_i(t) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e(t-\tau_i) \\ p(t-\tau_i) \end{bmatrix} \right\} . \tag{3-8}$$

Now let $x(t) = \begin{bmatrix} e(t) \\ p(t) \end{bmatrix}$, $A = \begin{bmatrix} 0 & 0 \\ m & 0 \end{bmatrix}$ and $B_i(t) = \begin{bmatrix} n_i(t) & n_i(t) \\ 0 & 0 \end{bmatrix}$, and write Equation (3-8) as

$$\dot{x}(t) = Ax(t) + \sum_{i=1}^{N} B_i(t)x(t-\tau_i) . \tag{3-9}$$

The system of Equation (3-9) is a first-order time delay control system. We take two steps to prove its stability. First, we prove in Theorem 3-1 the system can be asymptotically stable if the time delay does not exist. Then, we prove in Theorem 3-2 the stability of the system with time delay and give the stability criteria.

3.2.1 Zero-Time-Delay System

In a zero-time-delay system, all $\tau_i = 0$, $i = 1, \cdots, N$. Then we obtain the system in Equation (3-9) as

$$\dot{x}(t) = \left[A + \sum_{i=1}^{N} B_i(t) \right] x(t)$$

$$= \begin{bmatrix} \sum_{i=1}^{N} n_i(t) & \sum_{i=1}^{N} n_i(t) \\ m & 0 \end{bmatrix} x(t)$$

$$= \begin{bmatrix} -n & -n \\ m & 0 \end{bmatrix} x(t)$$

$$= Cx(t) \quad . \tag{3-10}$$

where $C = \left[A + \sum_{i=1}^{N} B_i(t) \right] = \begin{bmatrix} -n & -n \\ m & 0 \end{bmatrix}$ and $n = \sum_{i=1}^{N} k a_i(t - \tau_{i1})$.

In a real network environment, the number of active connections has been shown experimentally to be proportional to the maximum connections N [Zhu02]. Therefore we can let $n = \sum_{i=1}^{N} k a_i(t - \tau_{i1}) = \beta N k$, where β is a positive constant and $0 \le \beta \le 1$.

Theorem 3-1 *The closed-loop system of Equation (3-10) is asymptotically stable if* $k > 0$ *and* $d > 0$.

Proof:

We want to find whether or not there exists a positive-definite symmetric matrix $R = \begin{bmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{bmatrix}$ such that

$$C^T R + RC = -I \quad , \tag{3-11}$$

where I is the identity matrix. If R exists, then we know from Lypunov Theorem [Ogat90] that the system is asymptotically stable.

Now Equation (3-11) can be written in matrix as

$$\begin{bmatrix} -n & m \\ -n & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{bmatrix} + \begin{bmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{bmatrix} \begin{bmatrix} -n & -n \\ m & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad . \tag{3-12}$$

By expanding the above matrix equation, we obtain three simultaneous equations as follows.

$$\begin{cases} 2nr_{12} = 1 \\ 2nr_{11} - 2mr_{12} = 1 \\ nr_{11} + nr_{12} - mr_{22} = 0 \end{cases} \quad . \tag{3-13}$$

Solving for r_{11}, r_{12} and r_{22}, we obtain

$$\begin{bmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{bmatrix} = \begin{bmatrix} \frac{m}{2n^2} + \frac{1}{2n} & \frac{1}{2n} \\ \frac{1}{2n} & \frac{1}{m} + \frac{1}{2n} \end{bmatrix} \quad . \tag{3-14}$$

It is easy to see that if $k > 0$ and $d > 0$, then $m > 0$, $n > 0$ and the determinants of the successive principal minors in matrix R are positive. Hence R is positive definite, and the equilibrium state of the system of Equation (3-10) is asymptotically stable [Ogat90, Chapter 9].

Q.E.D.

3.2.2 Non-Zero-Time-Delay System

Now we consider the system with non-zero time delay (Equation (3-9)) and derive its stability criteria. We can rewrite Equation (3-9) as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + \sum_{i=1}^{N} B_i(t)x(t) - \left[\sum_{i=1}^{N} B_i(t)x(t) - \sum_{i=1}^{N} B_i(t)x(t-\tau_i)\right] \\ &= Cx(t) - \sum_{i=1}^{N} B_i(t)[x(t) - x(t-\tau_i)] \\ &= Cx(t) - \sum_{i=1}^{N} B_i(t)\varphi_i(t) \quad , \end{aligned} \tag{3-15}$$

where $\varphi_i(t) = x(t) - x(t-\tau_i)$.

From Theorem 3-1, we know there exists a Lyapunov function

$$V(x(t)) = x^T(t)Rx(t) \quad , \tag{3-16}$$

for the zero-time-delay system of Equation (3-10) and it satisfies the following conditions:

$$\dot{V}(x(t)) = -x^T(t)x(t) = -\|x(t)\|^2 \quad , \tag{3-17}$$

and

$$\|\partial V/\partial x\| = \|2Rx(t)\| \le 2\|R\| \cdot \|x(t)\| \quad . \tag{3-18}$$

Because R is positive-definite, we know that the Lyapunov function of Equation (3-16) would satisfy any state vector. Therefore, when used for the non-zero time delay system, the following equation is also true:

$$V(x(t)) = x^T(t)Rx(t) > 0 \quad . \tag{3-19}$$

Let $\|\varphi(t)\|_{\max} = \max\{\|\varphi_i(t)\|, i = 1,\ldots,N\}$ and $D = \begin{bmatrix} -n & -n \\ 0 & 0 \end{bmatrix}$. If we consider the term $\sum_{i=1}^{N} B_i(t)\varphi_i(t)$ in Equation (3-15) to be a generalized vanishing perturbation [Khal96], then the derivative of $V(x(t))$ along the trajectories of the perturbed system satisfies

$$\begin{aligned} \dot{V}(x(t)) &\le -\|x(t)\|^2 + 2\|R\| \cdot \|x(t)\| \cdot \left\| \sum_{i=1}^{N} B_i(t)\varphi_i(t) \right\| \\ &\le -\|x(t)\|^2 + 2\|R\| \cdot \|x(t)\| \cdot \left\| \sum_{i=1}^{N} B_i(t) \right\| \cdot \|\varphi(t)\|_{\max} \\ &= -\|x(t)\|^2 + 2\|R\| \cdot \|x(t)\| \cdot \|D\| \cdot \|\varphi(t)\|_{\max} \quad . \end{aligned} \tag{3-20}$$

In our studied system, the time delay is quite small. If we let $\tau_{\max} = \max\{\tau_i, i = 1,\ldots,N\}$, then we have the following derivation based on Equation (3-15).

$$\begin{aligned} \|\varphi(t)\|_{\max} &= \|\Delta x(t)\|_{\max} \\ &\cong \left\| \dot{x}(t)\Delta t_{\max} \right\| \\ &= \left\| \dot{x}(t)\tau_{\max} \right\| \\ &= \left\| \tau_{\max} Cx(t) - \tau_{\max} \sum_{i=1}^{N} B_i(t)\varphi_i(t) \right\| \\ &\le \|\tau_{\max} Cx(t)\| + \left\| \tau_{\max} \sum_{i=1}^{N} B_i(t)\varphi_i(t) \right\| \\ &\le \tau_{\max}\|C\| \cdot \|x(t)\| + \tau_{\max}\|D\| \cdot \|\varphi(t)\|_{\max} \quad . \end{aligned} \tag{3-21}$$

Then we have

$$\left[1 - \tau_{\max}\|D\|\right] \cdot \|\varphi(t)\|_{\max} \le \tau_{\max}\|C\| \cdot \|x(t)\| \quad . \tag{3-22}$$

We shall consider the following two cases:

Case (i): $1-\tau_{\max}\|D\|>0$. (3-23)

Rearranging Equation (3-22), we obtain

$$\|\varphi(t)\|_{\max} \le \frac{\tau_{\max}\|C\|}{1-\tau_{\max}\|D\|}\cdot\|x(t)\| \quad . \tag{3-24}$$

By substituting $\|\varphi(t)\|_{\max}$ in Equation (3-20), we have

$$\begin{aligned}\dot{V}(x(t)) &\le -\|x(t)\|^2 + 2\|R\|\cdot\|x(t)\|\cdot\|D\|\cdot\frac{\tau_{\max}\|C\|}{1-\tau_{\max}\|D\|}\cdot\|x(t)\| \\ &= -\|x(t)\|^2 + 2\|R\|\cdot\|D\|\cdot\frac{\tau_{\max}\|C\|}{1-\tau_{\max}\|D\|}\cdot\|x(t)\|^2 \\ &= \gamma\cdot\|x(t)\|^2 \quad , \end{aligned} \tag{3-25}$$

where $\gamma = 2\|R\|\cdot\|D\|\cdot\frac{\tau_{\max}\|C\|}{1-\tau_{\max}\|D\|}-1$.

It is easy to get for any norm space

$$\begin{aligned}\|R\| &\le \left|\frac{m}{2n^2}+\frac{1}{2n}\right|+\left|\frac{1}{m}+\frac{1}{2n}\right| \\ &\le \left|\frac{m}{2n^2}\right|+\left|\frac{1}{n}\right|+\left|\frac{1}{m}\right| \quad , \end{aligned} \tag{3-26}$$

$$\|C\| \le \left|\sqrt{2}n\right|+|m| \quad , \tag{3-27}$$

and

$$\|D\| \le \left|\sqrt{2}n\right| \quad . \tag{3-28}$$

Then we have

$$\begin{aligned}\gamma &\le 2\left(\left|\frac{m}{2n^2}\right|+\left|\frac{1}{n}\right|+\left|\frac{1}{m}\right|\right)\cdot\left|\sqrt{2}n\right|\frac{\tau_{\max}\left(\left|\sqrt{2}n\right|+|m|\right)}{1-\tau_{\max}\left|\sqrt{2}n\right|}-1 \\ &= \sqrt{2}\left(\frac{m}{n}+2+\frac{2n}{m}\right)\cdot\frac{\tau_{\max}(\sqrt{2}n+m)}{1-\sqrt{2}\tau_{\max}n}-1 \\ &= \left(\frac{m}{n}+2+\frac{2n}{m}\right)\cdot\frac{\sqrt{2}+m/n}{1/\sqrt{2}\tau_{\max}n-1}-1\end{aligned}$$

$$= \left(\frac{sd}{\beta Nk} + 2 + \frac{2\beta Nk}{sd}\right) \cdot \frac{\sqrt{2} + \frac{sd}{\beta Nk}}{\frac{1}{\sqrt{2}\beta Nk\tau_{\max}} - 1} - 1 \quad . \tag{3-29}$$

Case (ii): $1 - \tau_{\max}\|D\| \le 0$. (3-30)

Following a development similar to that from Equation (3-23) to Equation (3-29), we can obtain

$$\gamma \le \left(\frac{sd}{\beta Nk} + 2 + \frac{2\beta Nk}{sd}\right) \cdot \frac{\sqrt{2} + \frac{sd}{\beta Nk}}{1 - \frac{1}{\sqrt{2}\beta Nk\tau_{\max}}} - 1 \quad . \tag{3-31}$$

Because $1 - \frac{1}{\sqrt{2}\beta Nk\tau_{\max}} > 0$ and all the items in Equation (3-31) are positive, it is easy to see that it is impossible for $\gamma < 0$ with Equation (3-31).

Theorem 3-2 *The time delay closed-loop system of Equation (3-9) is asymptotically stable if*

$$0 < k < \frac{1}{2N\tau_{\max}} \quad , \tag{3-32}$$

$$0 < d \quad , \tag{3-33}$$

and

$$\left(\frac{sd}{\beta Nk} + 2 + \frac{2\beta Nk}{sd}\right) \cdot \frac{\sqrt{2} + \frac{sd}{\beta Nk}}{\frac{1}{\sqrt{2}\beta Nk\tau_{\max}} - 1} - 1 < 0 \quad . \tag{3-34}$$

Proof:

If the conditions of Equation (3-32) to (3-34) hold, then the assumption of the case (i) (Equation (3-23)) is satisfied and that case is true. From the above derivation, it can be shown that

$$V(x(t)) > 0 \text{ and } \dot{V}(x(t)) < 0 \quad .$$

Hence, according to Lyapunov Second Stability Criterion, the closed-loop system is asymptotically stable.

Q.E.D.

3.3 Implementation of The PIR Controller

We use two different ways to implement our PIR controller. The first implementation needs to know the real RTT (round-trip time) value, and is called PIR-A in this thesis. The second one uses a roughly pre-set fixed value for RTT, and is called PIR-B.

To make the implementation of the PIR controller simpler, we apply a simplified version of Equation (3-2) as follows. Though it is simpler than that in Equation (3-2), we have seen it can achieve almost the same capability.

$$q_{sp}(t) = \begin{cases} q_{sp}(t^-) + d, & if \quad q(t) < q_{SP} \\ q_{sp}(t^-), & if \quad q(t) = q_{SP} \\ q_{sp}(t^-) - d, & if \quad q(t) > q_{SP} \end{cases} . \tag{3-35}$$

3.3.1 PIR-A Implementation

We use the window-based calculation to convert the recommended global source rate into window size for the PIR controller. The advertised window size $win_i^{ad}(t)$, $i = 1,2,\cdots,N$ is computed as

$$\begin{aligned} win_i^{ad}(t) &= rtt_i(t) \cdot \mu(t) \\ &= rtt_i(t)\{\mu^{\max} - k[q(t) - q_{sp}(t)]\} \end{aligned} , \tag{3-36}$$

where $rtt_i(t)$ is the varied round-trip time for Source i.

Figure 3-2 shows the algorithm of PIR-A. When a packet sent from the TCP source is received in the router, it brings with it the value of the current round-trip time in a field added to the TCP header. The router then computes the advertised window size according to Equation (3-36). Next, the receiver window field in the packet TCP header is updated with the value obtained from Equation (3-36) before the packet being relayed to its destination. The destination extracts the advertised window size from the received packet, copies it into

the receiver window field of the ACK TCP header and then sends the ACK back to the appropriate

Router:

for each packet arrival
    if q >= buffer size
        discard the packet
    else
        retrieve rtt from the TCP header
        update q_{sp} :
            if $q < q_{SP}$ then $q_{sp} = q_{sp} + d$
            if $q = q_{SP}$ then $q_{sp} = q_{sp}$
            if $q > q_{SP}$ then $q_{sp} = q_{sp} - d$
        compute the advertised window size:
            $win^{adv} = rtt[\mu_{\max} - k(q - q_{sp})]$
        update the window field in the TCP header
        send out the packet

Destination:

for each packet arrival
    retrieve the advertised window size from the TCP header
    copy the advertised window size into the ACK TCP header
    send out the ACK

Source:

for each ACK arrival
    retrieve the advertised window size from the TCP header
    update the window size:
        $win = win^{adv}$

Saved Variables:

q_{sp} : varied queue size setpoint
win : window size of TCP source

Parameters:

q_{SP} : reference queue size
d : adaptation constant of PIR-A
k : proportional gain of PIR-A
$\mu_{\max}$: recommended peak source rate

Other:

q : instantaneous queue size
rtt : link round-trip delay
win^{adv} : advertised window size

Figure 3-2 Algorithm of PIR-A Implementation

Destination:

for each packet arrival

update the receiver window size in the ACK TCP header

send out the ACK

Router:

for each ACK arrival

update q_{sp} :

if $q < q_{SP}$ then $q_{sp} = q_{sp} + d$

if $q = q_{SP}$ then $q_{sp} = q_{sp}$

if $q > q_{SP}$ then $q_{sp} = q_{sp} - d$

compute the advertised window size:

$$win^{adv} = rtt[\mu_{\max} - k(q - q_{sp})]$$

update the window field in the ACK TCP header

if $win^{adv} < win^{rcv}$ then $win^{rcv} = win^{adv}$

send out the ACK

Source:

for each ACK arrival

retrieve the receiver window size from the TCP header

update the window size:

$$win = \min(win^{rcv}, cwnd + dup\_acks)$$

Saved Variables:

q_{sp} : varied queue size setpoint

win : window size of TCP source

$cwnd$: congestion window size of TCP source

dup_acks : number of duplicate ACKs received in TCP sources

Parameters:

q_{SP} : reference queue size

d : adaptation constant of PIR-B

k : proportional gain of PIR-A

$\mu_{\max}$: recommended peak source rate

RTT : pre-set fixed round-trip delay

Other:

q : instantaneous queue size

win^{adv} : advertised window size

win^{rcv} : receiver window size

Figure 3-3 Algorithm of PIR-B Implementation

source. After receiving the ACK, the TCP source determines the data window size $win_i(t)$, $i = 1,2,\cdots,N$ as

$$win_i(t) = win_i^{adv}(t - \tau_{i2}) \quad . \tag{3-37}$$

3.3.2 PIR-B Implementation

Unlike PIR-A, PIR-B uses a fixed pre-set RTT to compute the advertised window size $win_i^{ad}(t)$, $i = 1,2,\cdots,N$ as Equation (3-38) shows.

$$\begin{aligned} win_i^{ad}(t) &= RTT \cdot \mu(t) \\ &= RTT\{\mu^{\max} - k[q(t) - q_{sp}(t)]\} \quad . \end{aligned} \tag{3-38}$$

Figure 3-3 shows the algorithm of PIR-B. We use a receiver window updating method similar to that in [AwOu02] for PIR-B. When the router receives a packet from the TCP source, it does not touch the receiver window field in the packet TCP header. It is until the ACK is received that the router updates the receiver window field in the ACK TCP header with the value obtained from Equation (3-38) if the value is less than the one that is already in the field. The TCP source still uses its current algorithm to determine its window size $win_i(t)$, $i = 1,2,\cdots,N$ (suppose the current TCP is Tahoe or Reno), e.g. Equation (3-39) for Reno.

$$win_i(t) = \min[win_i^{adv}(t - \tau_{i3}), cwnd + dup\_acks] \quad , \tag{3-39}$$

where τ_{i3} is the path delay from the router to Source i.

The value of the fixed pre-set RTT can be roughly chosen from a wide range. As we will see later from the simulation results, it has little effect on the system stability.

3.4 Concluding Remarks

Each of the PIR controllers has its advantages and disadvantages. Like that of ECN, the limitation of PIR-A is that it needs the support from both the router and the end node. Since most of the current TCP/AQM schemes are Reno/RED, PIR-A requires the implementation be changed for both the router and the end node. However, PIR-A can be a choice for network traffic congestion control in the future. On the other hand, PIR-B only needs the

support of the router, but it will not work well if the packet path and the ACK path for one connection do not go through the same bottleneck router, so it is suitable for network areas where the packets and ACKs for a TCP source have the same bottleneck router, e.g. the network edges (of ISPs and enterprise networks).

As we have seen, the PIR controller does not compute the advertised window sizes for all the flows at the same time, and these sizes are different from each other if the round-trip delay for each flow is different. So the global synchronization problem will not occur.

Though the parameter tuning for our PIR controller is not difficult, it is not a trivial job, either. We will provide the PIR controller design guideline in details in Chapter 6, e.g. how to choose the values for the reference queue size, the global proportional gain and the adaptation constant. Besides, we also suggest a method to implement the PIR controller in a multiple-router environment in Chapter 6.

Chapter 4
Performance Evaluation in Wired Network

In this chapter, we will study via simulations various performance issues of the PIR controllers in the wired network configuration. The network setup and purpose have been presented in Figure 2-1, and the queue control model in Figure 2-3.

4.1 Simulation

We shall verify our proposition via simulations using the OPNET simulator, and compare the simulation results of our PIR controller in both implementation cases (PIR-A and PIR-B) with those of RED, ECN, and PI-RED. The simulation measurements are taken from the bottleneck router (Router 1). The following four performance measures are used to do the comparison in the wired network.

(1) Instantaneous buffer queue size: this is defined to be the queue size sampled at each fixed interval of 1 second in the bottleneck router.

(2) Packet drop rate: this is defined to be the number of packets dropped divided by the time duration between two consecutive sampling instances in the bottleneck router.

(3) Average router throughput: this is defined to be the cumulative router throughput (in bits) so far divided by the time duration from the beginning of simulation to the current simulation time.

(4) Average queue size deviation: this is defined to be the time average of the absolute difference between the real queue size sampled and the reference queue size in the bottleneck router. We use the average queue size deviation metric to measure the level of the queue length stability. A smaller average queue size deviation indicates a better stability, whereas a larger deviation shows that the queue size stability is worse.

All network systems are simulated for a time period of 120 seconds, which we have determined to be adequate to establish reasonable steady-state results, with about 5500 packets per second on the average received in the router. In the figures for steady-state performance analysis, each individual point is the average of 4 different simulation runs. This

enabled us to have obtained 95% confidence interval for an error criterion ± 1 packet for the average queue size deviation measure.

The different delays for different subnets are listed in Table 4-1.

Table 4-1 Propagation Delays in Wired Network

Subnet	Propagation delay
1	20 ms
2	40 ms
3	60 ms
4	80 ms
5	100 ms

The parameters used in the think time probability distribution function of Equation (2-5) are set as $p_h = 0.4953916$, $p_l = 0.5046084$, $\pi_h = 1.0$, $\pi_l = 0.0245032$, $\eta_h = 1.243437$, $\eta_l = 3.252665$, and the parameters of the file length probability distribution function of Equation (2-6) are $\pi = 2190$ and $\alpha = 1.15066$. The buffer size of Router 1 in the wired network is 800 packets. All the packets have the same size of 1024 bytes.

Based on the wired network model, we create four simulation scenarios, each with different number of ftp sources and http sources (see Table 4-2).

Table 4-2 Number of Ftp and Http Sources in Different Simulation Scenarios for Wired Network

Scenario No.	Number of ftp sources	Number of http sources	Network Configuration
1	100	0	Wired
2	75	25	Wired
3	50	50	Wired
4	25	75	Wired

As shown in Table 4-2, all the 100 TCP sources are ftp sources in Scenario 1, but their active periods are different. The first 75 ftp sources (15 for each subnet) start to send packets from the beginning of simulation. The other 25 ftp sources start to send packets from the 40^{th} second after simulation begins and stop transmission from the 80^{th} second. There are 75 ftp sources and 25 http sources in Scenario 2, 50 ftp sources and 50 http sources in Scenario 3,

and 25 ftp sources and 75 http sources in Scenario 4. All ftp and http sources in Scenarios 2 to 4 start to send packets from the beginning of simulation.

Table 4-3 Simulation Parameters for Wired Network

RED/ECN	
minimum buffer threshold	150
maximum buffer threshold	600
maximum packet drop probability	0.1
queue averaging filtering gain	0.002
PI-RED	
reference queue size	100
round-trip time limit	0.13
load factor	100
sample frequency	1/RTT
PIR-A	
reference queue size	100
recommended peak source rate	80
proportional gain	0.08
adaptation constant	0.5
PIR-B	
reference queue size	100
recommended peak source rate	80
proportional gain	0.08
adaptation constant	0.5
fixed *RTT*	0.06

To allow a fair comparison, we have tested quite a few sets of parameters according to the design guidelines for RED, ECN and PI-RED. The best performance we have seen from them [Zhu02, HoMi01b] is used here to do comparison. However, the average results have the same conclusions, e.g. PIR performs better than PI-RED. Table 4-3 lists the parameters for RED, ECN, PI-RED and our PIR controllers – PIR-A and PIR-B.

4.2 Performance Evaluation

The simulation results for Scenarios 1, 2, 3 and 4 are shown as follows.

4.2.1 Scenario 1

Figures 4-1a to 4-1e show the buffer queue size of the bottleneck router (Router 1) in Scenario 1. It can be seen from Figure 4-1a that the queue length in the RED router fluctuates

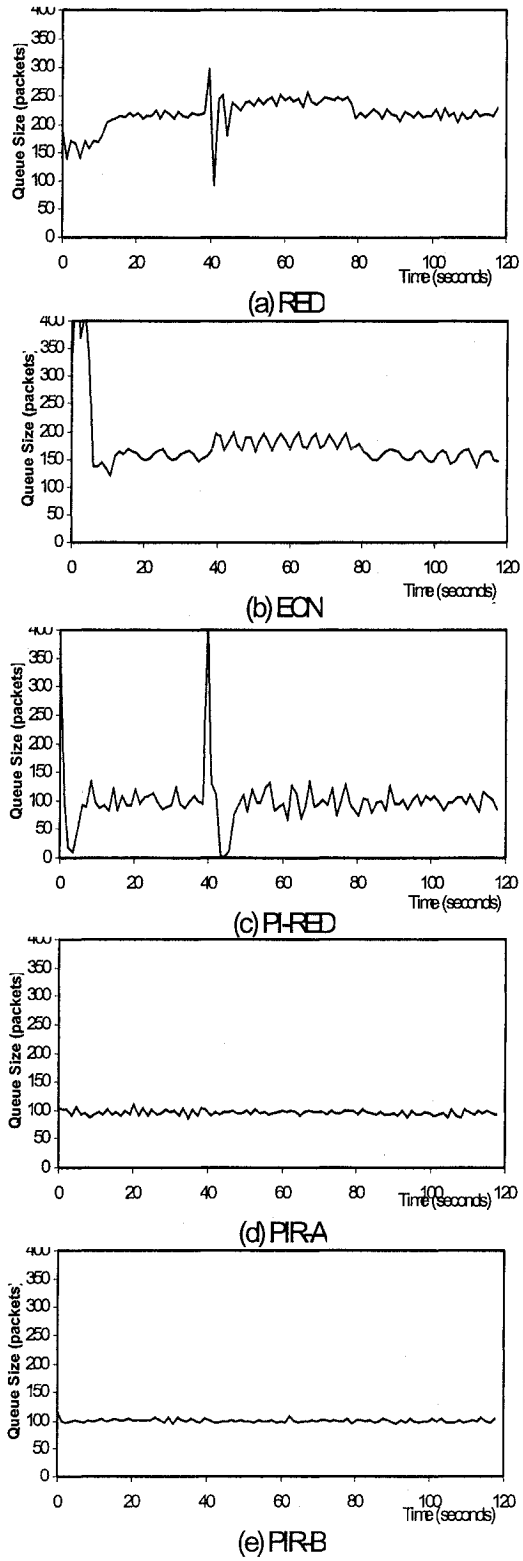


Figure 4-1 Evolution of Queue Size under Different Control for Scenario 1

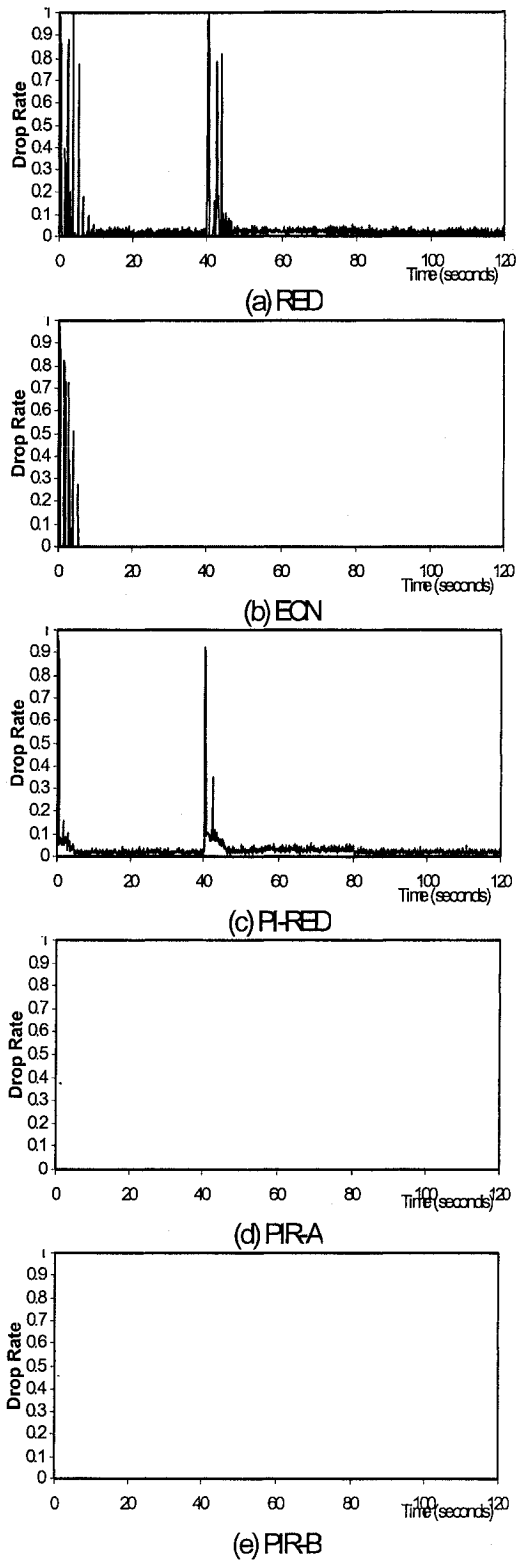


Figure 4-2 Packet Drop Rate under Different Control for Scenario 1

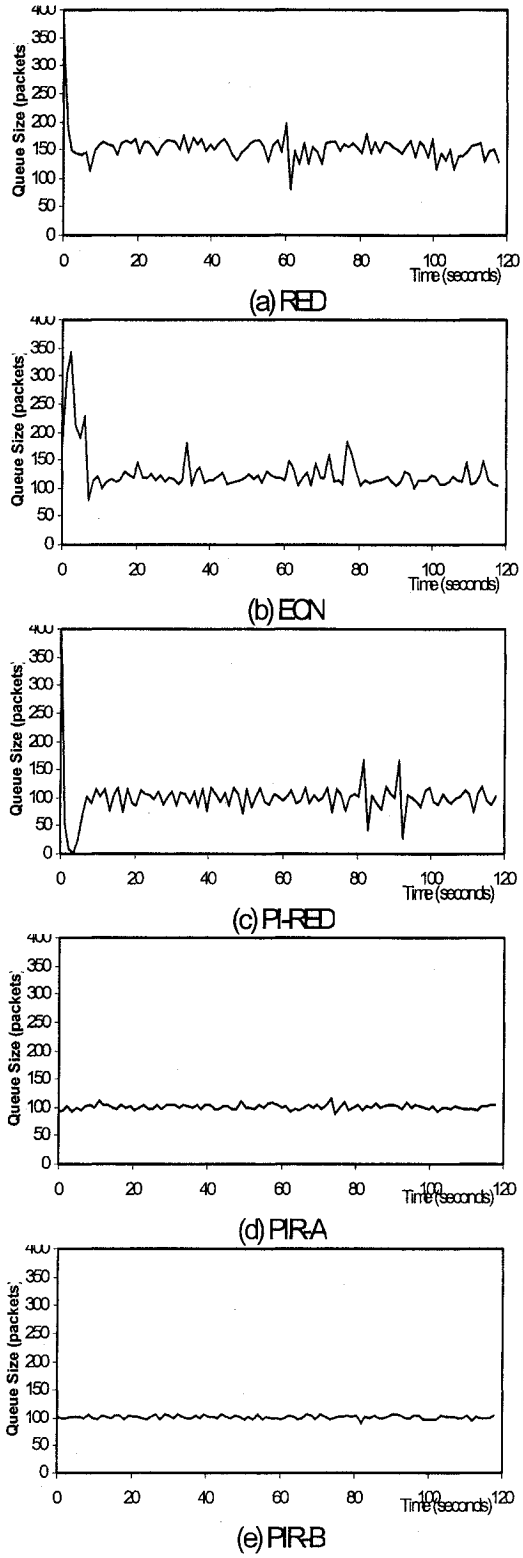


Figure 4-3 Evolution of Queue Size under Different Control for Scenario 2

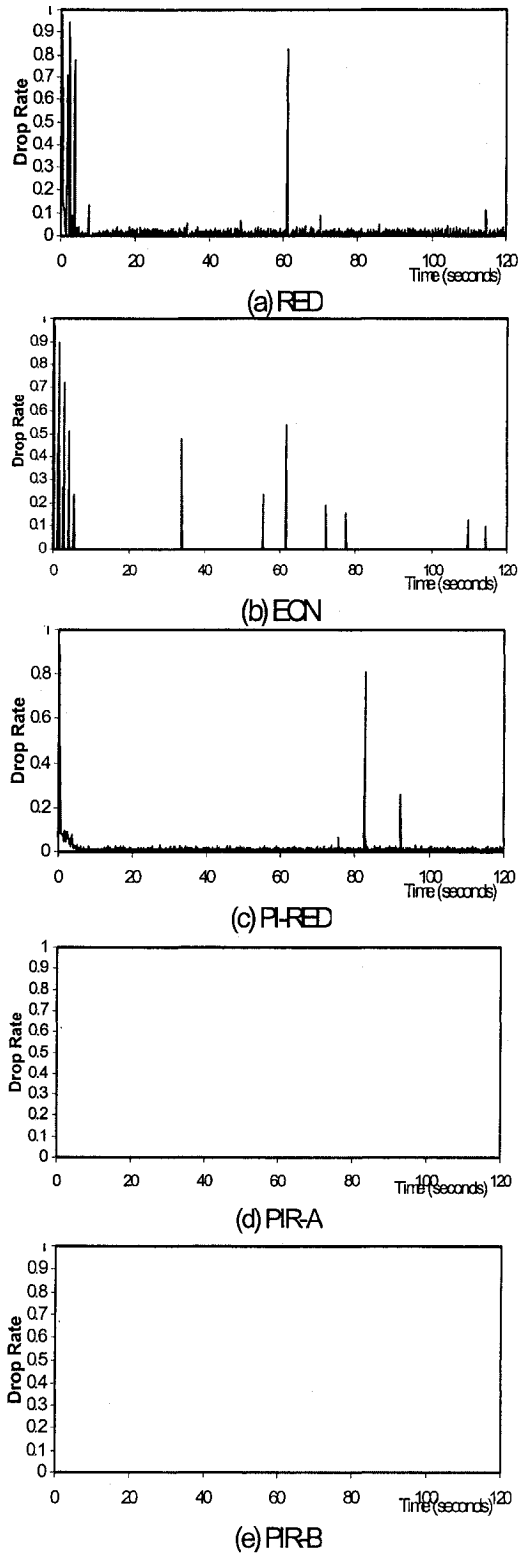


Figure 4-4 Packet Drop Rate under Different Control for Scenario 2

with a large amplitude. Though ECN can avoid the unnecessary packet loss, the ECN router also has the unstable queue size, as can be seen from Figure 4-1b. ECN also has a high initial queue size because no packets are dropped if the router buffer is not full and all the sources send packets aggressively in the beginning of simulation. In 4-1a and 4-1b, we can see that the queue size is larger during the time period 40 – 80 seconds. This is because there are 100 active ftp sources in the period of 40 – 80 seconds, and there are only 75 active ftp sources in other time. Although PI-RED can keep its queue size around 100 (Figure 4-1c), the pre-set reference queue size, its queue size oscillation is still noticeable, especially at the 40^{th} second, when the number of active ftp sources rise up to 100 from 75. In Figure 4-1d and 4-1e, we can see that, under the regulation of our PIR controller, the queue size remains at the reference size level quite stably.

Figures 4-2a to 4-2e show the packet drop rate of Router 1 in Scenario 1. It can be seen from Figure 4-2a that the RED router always drops packets in order to stay uncongested. The highest packet drop rate appears at the 0^{th} second and the 40^{th} second, when most of the packets entering the router (for different reasons) are dropped. In the beginning ($t=0$), all the sources start to send packets to the router. By the time the congestion information is sent back to the source, the router has already received far more packets than it can process. Consequently, most of those packets have to be dropped. On the other hand, most of the packets at the 40^{th} second are from the 25 new ftp sources that enter the network simultaneously. As a result, many packets are dropped.

ECN behaves similarly to RED in the beginning of simulation. As can be seen from Figure 4-2b, many packets are dropped in the start time when packets arrive but the buffer is full. However, in the remaining time, no more packets are dropped because ECN does not drop packets to indicate the congestion status as RED does. From Figure 4-1c, we can see that PI-RED has the same behavior as RED except that the packet drop rate is smaller than the latter. That is because PI-RED also notify the sources of network congestion by dropping received packets but the drop probability is controlled with a more efficient way.

It is interesting to see that no packets are dropped from the bottleneck router that is controlled by the PIR controller (Figure 4-2d and 4-2e). Thus our PIR controller can achieve network stability without the sacrifice of packet losses.

Figures 4-9a shows the average throughput of Router 1 in Scenario 1. It can be seen that the RED router has a relatively lower throughput in the beginning. This is because the queue size oscillates severely and many packets are dropped in the beginning, which causes TCP sources to suffer in the packet transmission rate. After that, the average throughput gets high and becomes stable. ECN and PI-RED behave similarly to RED. In the beginning of simulation, they also have a lower and unstable throughput due to the same reason as that of RED. Then, next, the average throughput gets high and becomes stable. PI-RED has a short time period of throughput decrease at the 40^{th} second because of its reaction to the entering of 25 new ftp sources. PIR-A and PIR-B, compared with RED, ECN and PI-RED, show an outstanding performance. They achieve a high average throughput in Router 1 from the beginning and maintain this high throughput stable all through the simulation process.

4.2.2 Scenario 2

Figures 4-3a to 4-3e show the buffer queue size of the bottleneck router (Router 1) in Scenario 2. It can be seen from Figure 4-3a and 4-3b that the queue sizes of RED and ECN are relatively smaller, compared with those in Scenario 1, because some connections are short-lived flows (http). They send packets sporadically, which exposes lighter burden to the bottleneck router. From Figure 4-3a to 4-3e, we can see that, when there are http sources in the network, the queue size in the router becomes less stable for RED, ECN and PI-RED. However, the queue size is still stable for PIR-A and PIR-B.

Figures 4-4a to 4-4e show the packet drop rate of Router 1 in Scenario 2. Comparison observations similar to Scenario 1 are obtained. However, it can be seen that the number of dropped packets is smaller than that in Scenario 1 because most of the TCP sources are shorted-lived sources. The router receives far less packets from the sources. Hence, far less packets are dropped. For the same reason, we can see that packets are dropped more randomly for RED, ECN and PI-RED. In some time, when many http sources happen to start to send packets simultaneously, the packet drop rate can be very high (see some high drop rate points in Figure 4-4a to 4-4c). However, there is no packet drop for PIR-A and PIR-B.

Figures 4-9b shows the average throughput of Router 1 in Scenario 2, which is very similar to Scenrio1. PIR-A and PIR-B achieve almost the same high and stable throughput as in Scenario 1. The only difference we can see is that RED, ECN and PI-RED have a lower

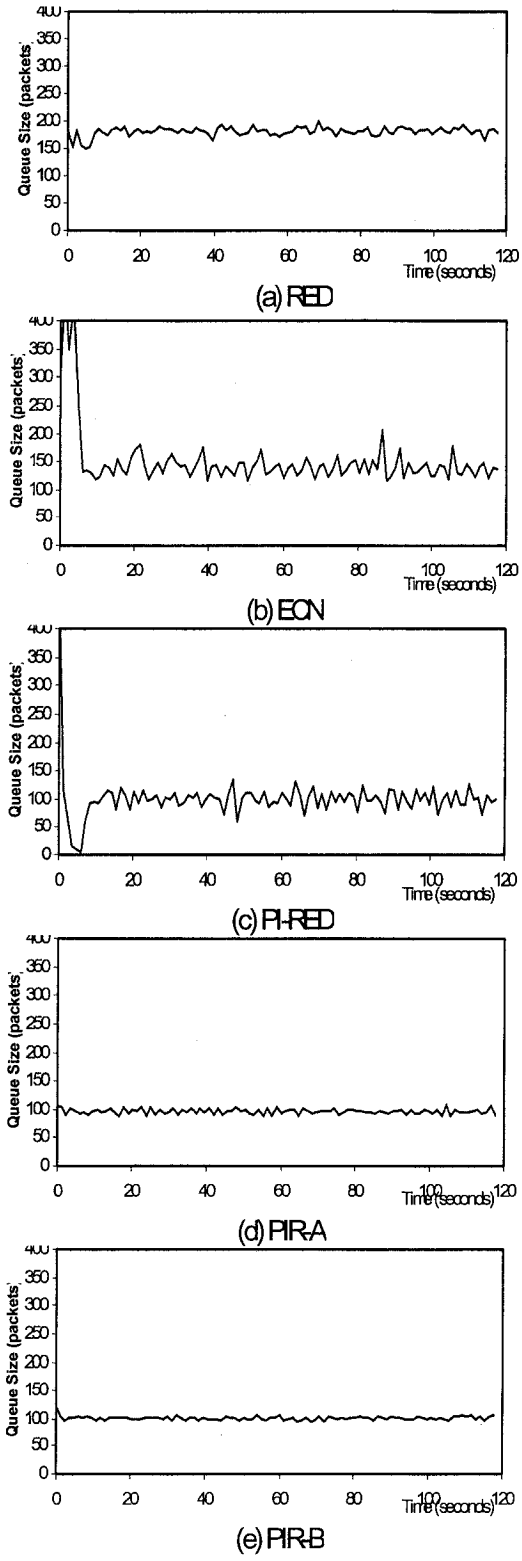


Figure 4-5 Evolution of Queue Size under Different Control for Scenario 3

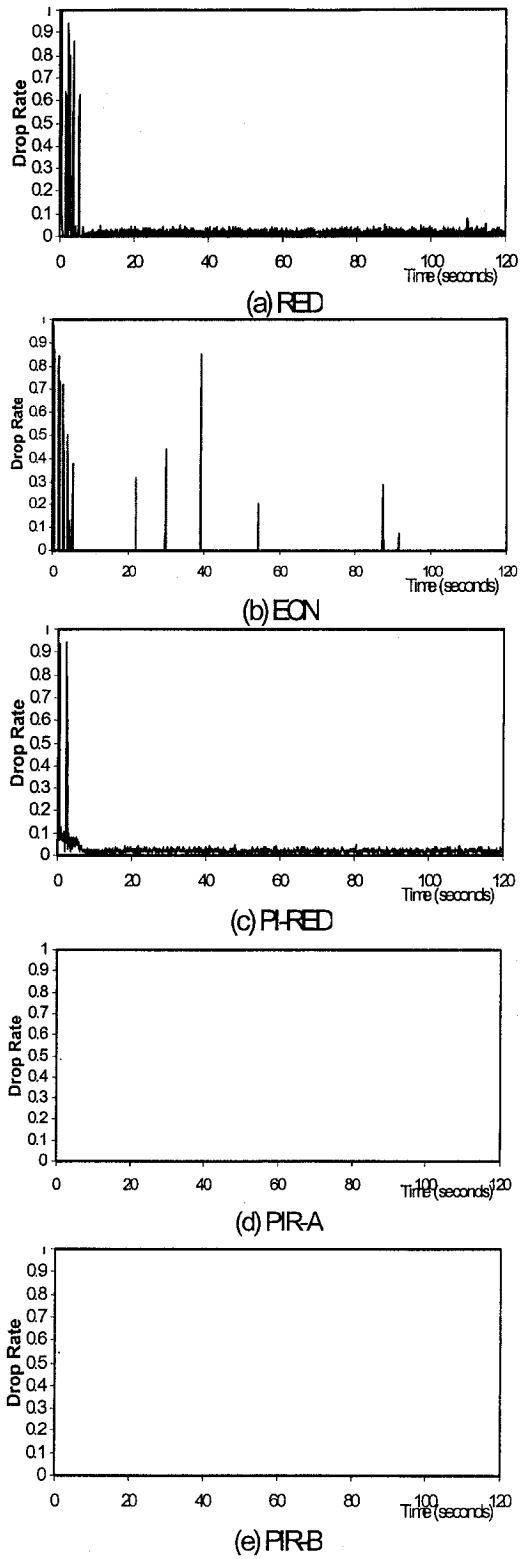


Figure 4-6 Packet Drop Rate under Different Control for Scenario 3

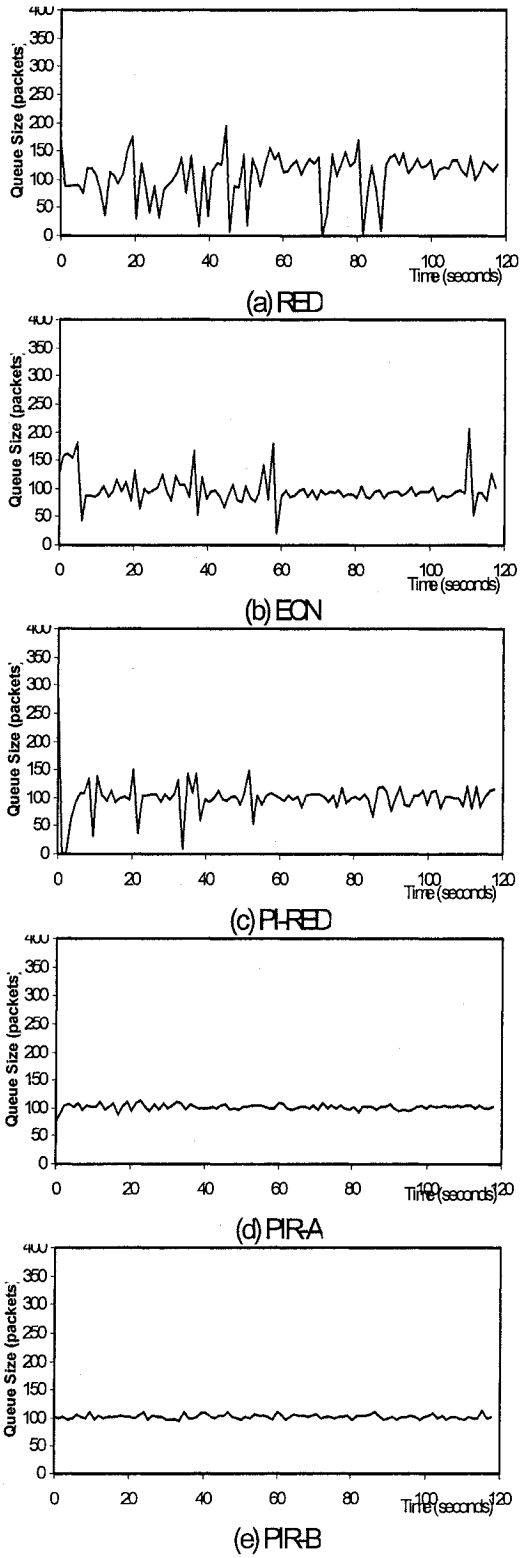


Figure 4-7 Evolution of Queue Size under Different Control for Scenario 4

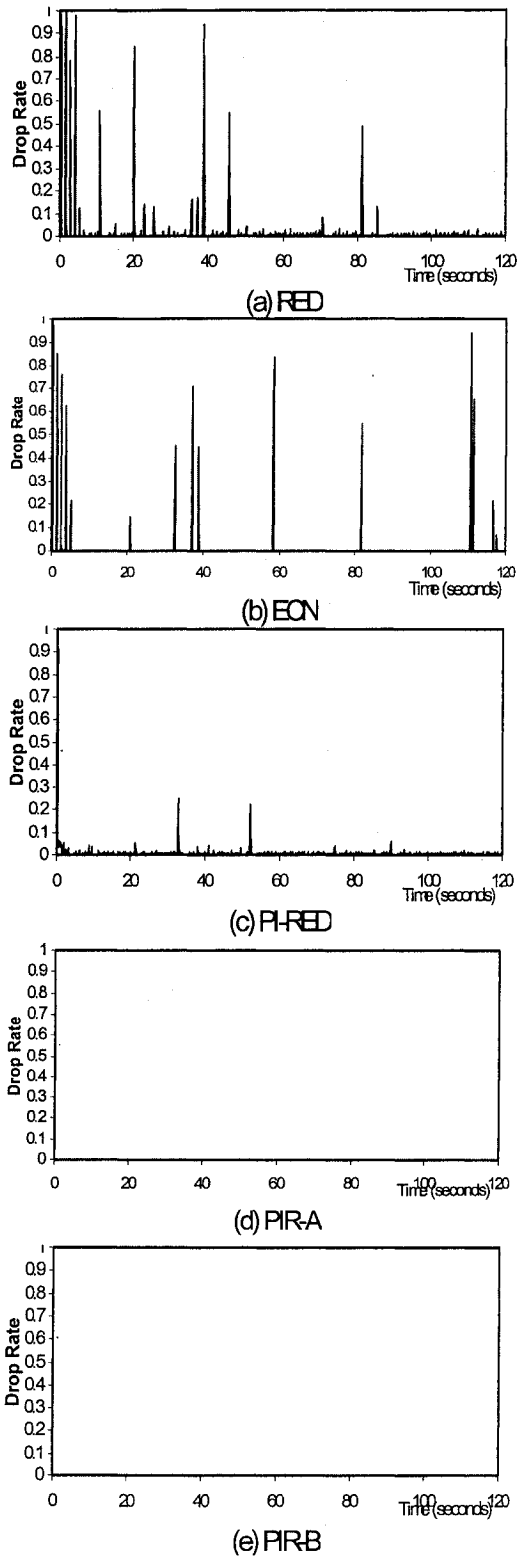


Figure 4-8 Packet Drop Rate under Different Control for Scenario 4

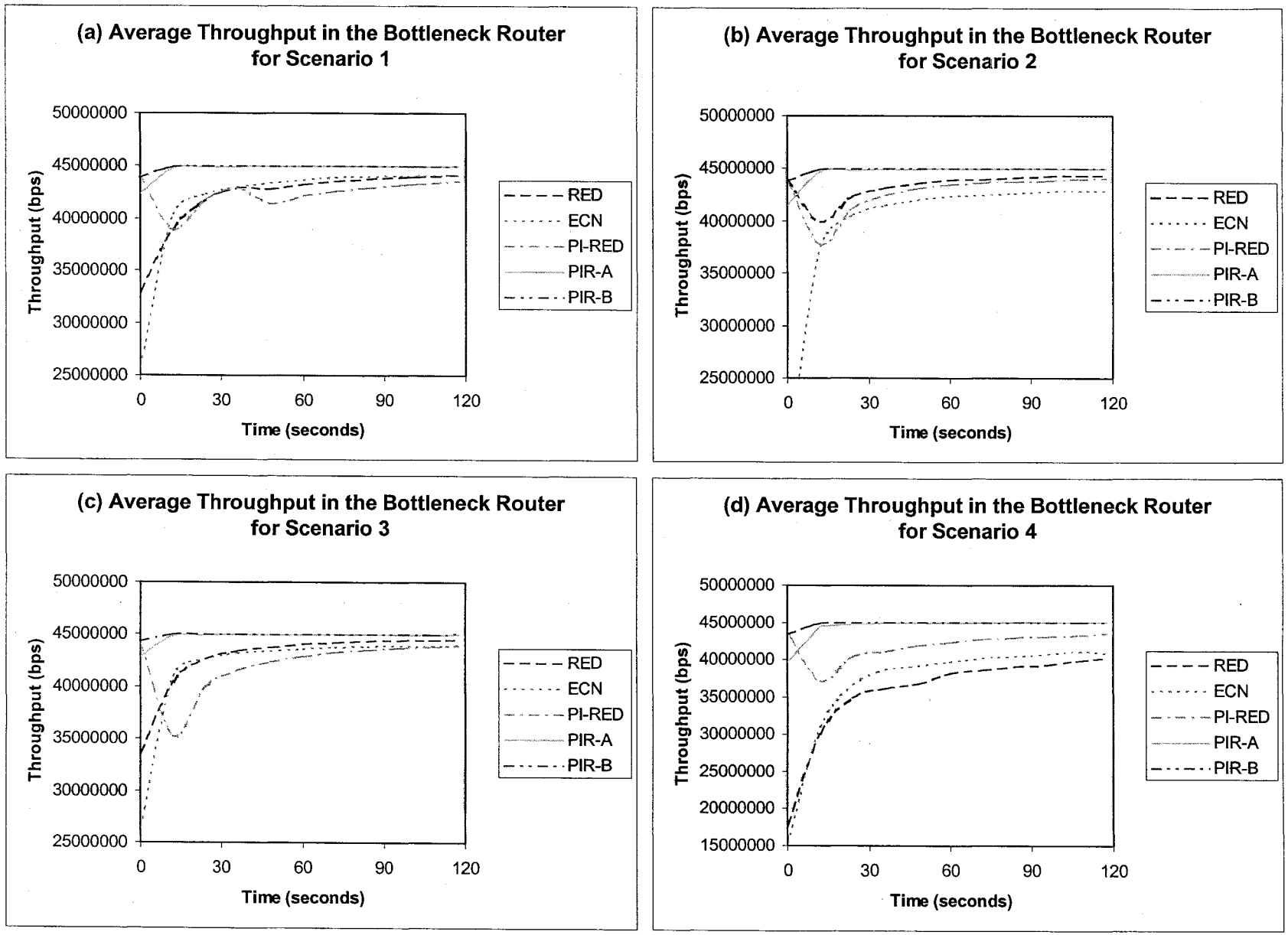


Figure 4-9 Average Throughput of the Bottleneck Router in the Wired Network for Different Scenarios

throughput than that in Scenario 1 because there are 25 http sources in Scenario 2. These http sources make the traffic through Router 1 controlled by RED, ECN or PI-RED oscillated more severely.

4.2.3 Scenarios 3 and 4

The queue length comparison, the packet drop comparison and the average router throughput comparison in Scenario 3 are shown in Figures 4-5a to 4-5e, Figures 4-6a to 4-6e, and Figure 4-9c, respectively. Likewise, the comparison in Scenario 4 is shown in Figures 4-7a to 4-7e, Figures 4-8a to 4-8e, and Figure 4-9d, respectively. As we can see, the performances of all the controllers in Scenarios 3 and 4 are similar to those in Scenario 2. It shows that the PIR controller can still work well in a network where most of the TCP sources are short-lived sources, which is more likely to happen in the real world.

As we know, the ratio of the number of http sources to the number of ftp sources rises from 0 to 1/3, 1, and 3, from Scenario 1 to Scenario 4. When this ratio becomes bigger, the queue size stability of RED, ECN and PI-RED becomes worse, because the increase of http sources cause more variance to the packet arrival rate in the router. As a result, the queue size in the bottleneck router oscillates more severely. Due to the same reason, the router throughput of RED, ECN and PI-RED becomes lower when the number of http sources increases. On the other hand, the packet drop rate of RED, ECN and PI-RED becomes smaller, when the number of http sources increases. This is because the packet arrival rate in the router becomes smaller, and hence fewer packets will be discarded. It is interesting to see that our PIR controllers can perform well in all the cases. They do not show any big difference in terms of queue size stability and packet drop rate from Scenario 1 to Scenario 4.

4.2.4 Effect of RTT

Figure 4-10 shows the average queue size deviations of PIR-B in the wired network for different scenarios when the pre-set RTT of PIR-B is set to be 0.02, 0.1, 0.2 and 0.3, respectively. We can see that the system stability in Scenario 1 is not sensitive to the value change of the pre-set RTT. From Scenario 2 to Scenario 4, the system is more and more sensitive. The more ratio of http sources, the more sensitive the system. However, generally speaking, the system stability is not sensitive to the value change of the pre-set RTT. No PIR-A is shown because it updates the RTT continuously.

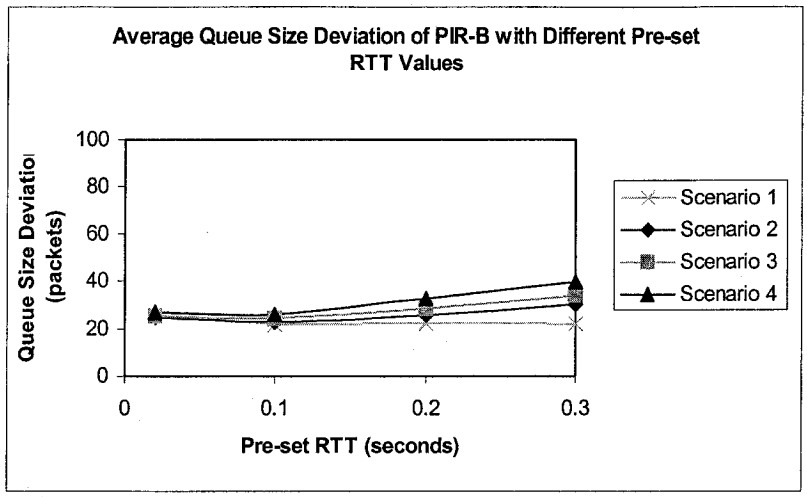


Figure 4-10 Average Queue Size Deviation of PIR-B in Wired Network with Different Pre-set RTT Values ($q_{SP} = 100$, $\mu_{max} = 80$, $k = 0.08$ and $d = 0.5$)

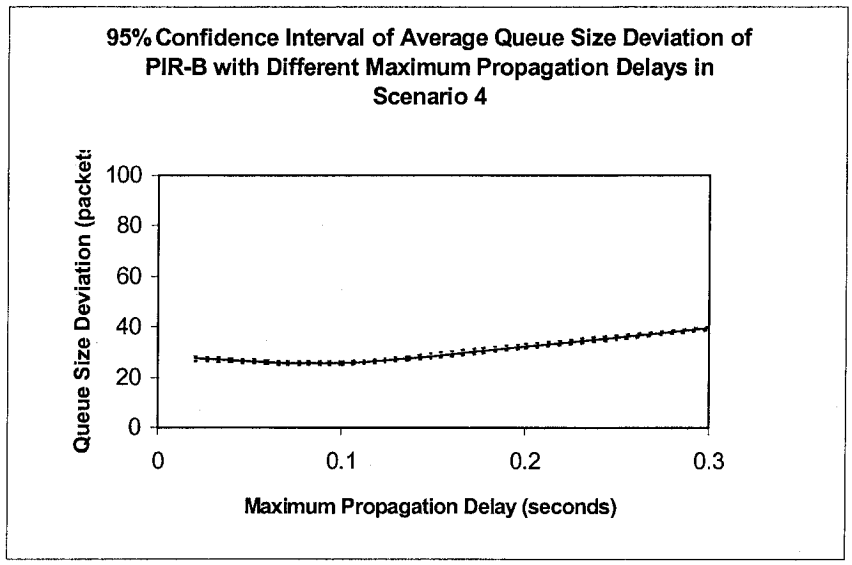


Figure 4-11 95% Confidence Interval of Average Queue Size Deviation of PIR-B in Wired Network with Different Pre-set RTT Values in Scenario 4 ($q_{SP} = 100$, $\mu_{max} = 80$, $k = 0.08$ and $d = 0.5$)

Figure 4-11 shows the 95% confidence interval of the average queue size deviations of PIR-B in the wired network in Scenario 4 when the pre-set RTT of PIR-B varies. As can be seen, the upper and lower confidence interval bounds are very close to the estimate of the average queue size deviations curve. Since all other curves have very similar observations, for clarity purpose, we just omit them for all the other curves in this thesis.

Though the maximum round-trip delay for a network can be estimated, there are chances that the maximum round-trip delay becomes much larger than that in the normal operation. In order to see how the PIR controller can stand the increase in the unexpected maximum round-trip delay, we have also conducted experiments by varying the maximum propagation delay in the network from 50ms, to 100ms, 150 ms, 200ms, 300ms, 400ms and 500 ms, while fixing all other network and simulation parameters.

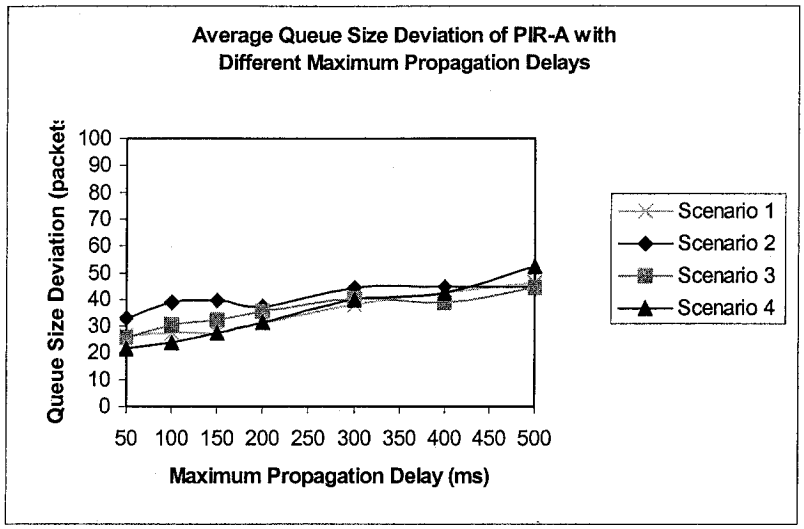


Figure 4-12 Average Queue Size Deviation of PIR-A in Wired Network with Different Maximum Propagation Delays (q_{SP} = 100, μ_{max} = 80, k = 0.08 and d = 0.5)

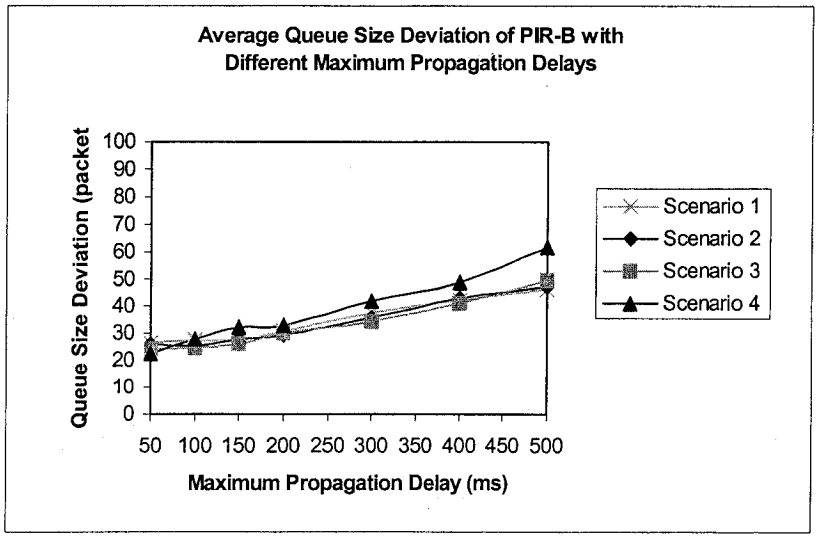


Figure 4-13 Average Queue Size Deviation of PIR-B in Wired Network with Different Maximum Propagation Delays (q_{SP} =100, μ_{max} =80, k = 0.08, d=0.5 and Pre-set RTT=0.06 sec)

Figure 4-12 shows the average queue size deviations of PIR-A in wired network when the maximum propagation delay is different. The performances of Scenarios 1, 2, 3 and 4 are all shown in the figure. We can see that, generally, when the maximum round-trip delay of the network becomes larger, the queue size becomes less stable for all the cases. The larger the maximum round-trip delay, the worse the queue size stability. However, we can notice some exceptions, when the maximum delay is around 100 to 150 ms. This is because the chosen controller parameters may not be the most suitable ones to the controller with the time delay. As we know from the stability criteria we proved in Chapter 3, the stability of the PIR controller is mainly determined by the time delay, and the parameter setting is affected by the time delay non-linearly. PIR-B has the similar behavior as PIR-A, as shown in Figure 4-13.

4.2.5 Effect of Proportional Gain k and Adaptation Constant d

We have known from the stability criteria proof in Chapter 3, in designing the PIR controller, whether the controller can be stable or not depends on how to choose the PIR controller proportional gain k and the adaptation constant d. In addition to the theoretical analysis of the controller stability criteria in Chapter 3, we also conducted a large number of simulation experiments with different combinations of k and d. The average queue size deviations in the bottleneck router (Router 1) in different cases are shown in Figure 4-14.

Figure 4-14a shows the average queue size deviation as a function of k for different d values under Scenario 1. In the case of d=0.02, the deviation exhibits a "valley" characteristic with high around k = 0.01 and k = 10 and a low at k = 0.1. Similar observations are obtained for d=0.1, 0.5, 1 and 5, but the "valley" value appears to shift from k = 0.1 to k = 0.01 as d increases from d = 0.02 to d = 1. The "valley" for d = 5 may also exist in the region of $k < 0.001$ in the figure. However, it is too small of a region to be seen in the figure. The "valley" phenomenon here can be explained with the stability criteria (Equation 3-32 to 3-34) we have developed in Section 3.2.2. It is easy to see from Equation 3-34 that, for a certain d value, there is a small range of k values that can satisfy the stability criteria. For k values out of the range, the system stability cannot be guaranteed.

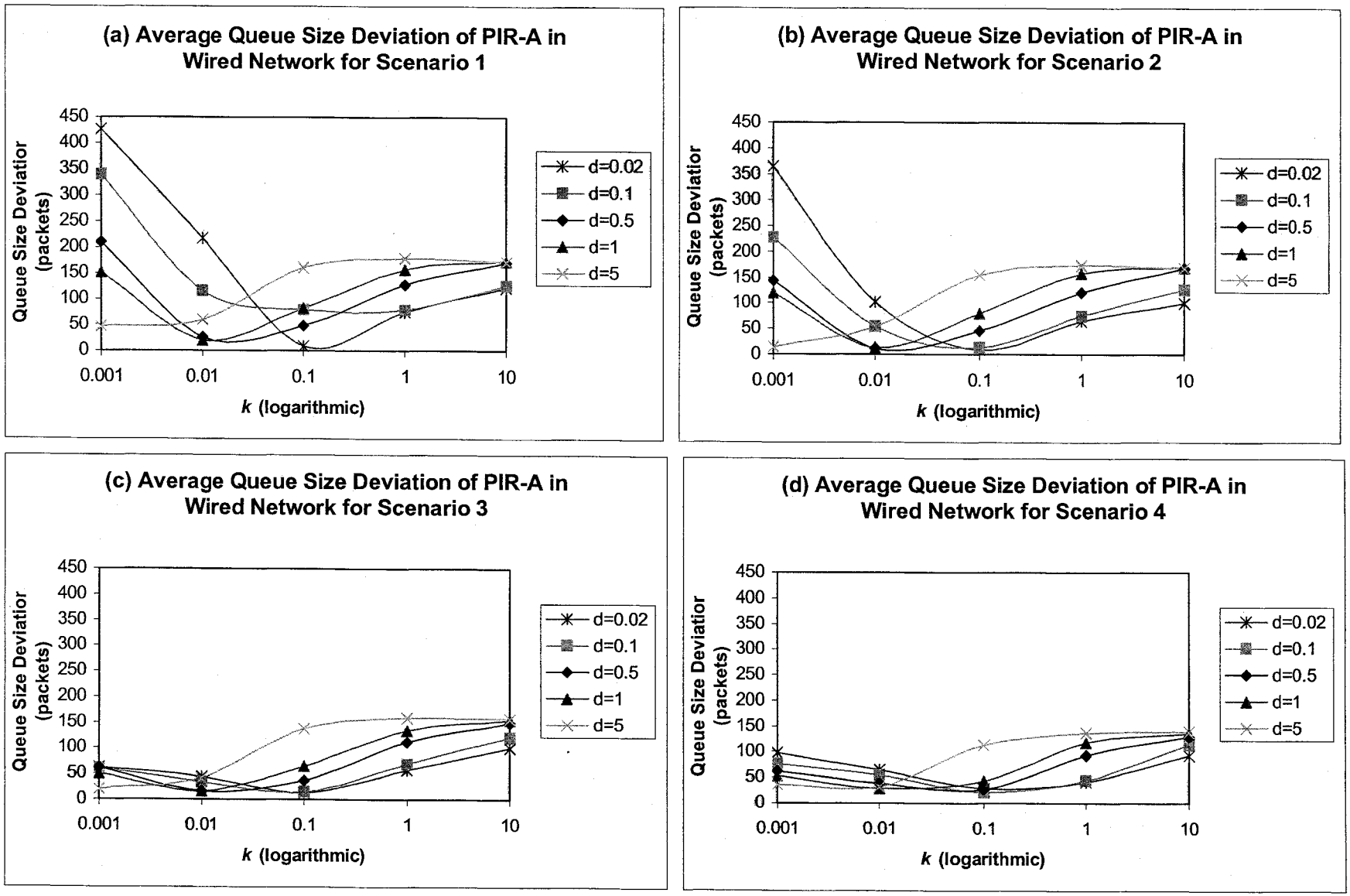


Figure 4-14 Average Queue Size Deviation under Different Combinations of PIR Controller Proportional Gain k and Adaptation Constant d in Wired Network ($q_{SP} = 100$, $\mu_{max} = 80$)

Figures 4-14b, 4-14c and 4-14d show similar performances to that of Figure 4-14a for Scenarios 2, 3 and 4. By comparing them with the performance of Scenario 1(Figure 4-14a), we can see that the queue size deviation becomes smaller when the ratio of the number of http sources to the number of ftp sources becomes bigger (from Scenario 1 to Scenario 4). This is because the more the ftp sources, the more the packets are received at the router, and the system oscillates more severely if the values of k and d are not set well. We used a wide range of the values for k and d to show the region of k and d which can perform best. For example, the combination of $k = 0.01$ and $d = 1$ is a proper setting for Scenario 1, whereas, the combination of $k = 0.001$ and $d = 1$ is not. The reason why the system oscillates severely with non-well-set values of k and d is that these values of k and d are far away from the stability region and hence cause the queue size to fluctuate with a big amplitude. As a result, for example, when k is set as 0.001 (the worst setting), the queue size deviation shows obvious increase from Scenario 4 to Scenario 3, Scenario 2 and Scenario 1. However, if the parameters of the controller are set properly, for example, when k is 0.1 and d is 0.5, no big difference of the performance is observed for the four different scenarios. PIR-B has the similar behavior as PIR-A.

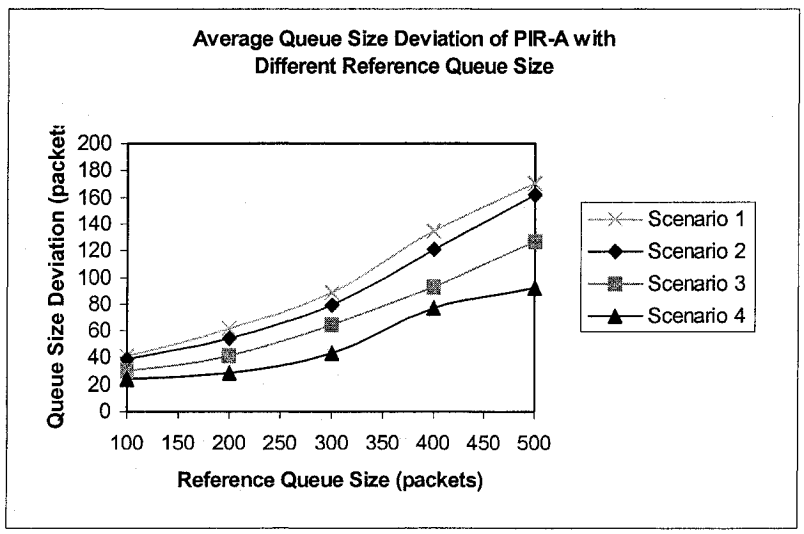


Figure 4-15 Average Queue Size Deviation of PIR-A in Wired Network with Different Reference Queue Size q_{SP} ($\mu_{max} = 80$, $k = 0.08$ and $d = 0.5$)

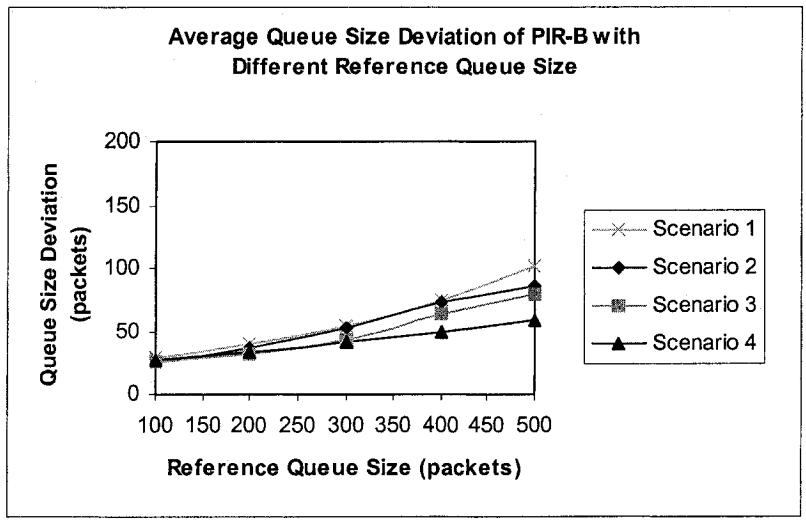


Figure 4-16 Average Queue Size Deviation of PIR-B in Wired Network with Different Reference Queue Size q_{SP} (μ_{max}=80, k = 0.08, d=0.5 and Pre-set RTT=0.06 sec)

4.2.6 Effect of Reference Queue Size q_{SP}

Figure 4-15 shows the average queue size deviations of PIR-A in different scenarios when the reference queue size q_{SP} varies. For Scenario 1, when q_{SP} is set to be 100, the smallest queue size deviation is observed. When q_{SP} varies from 200 to 300, 400 and 500, we can see that the queue size becomes less and less stable. This is because larger reference queue size causes larger queuing delay, which results in a larger round-trip delay. Thus, the system becomes less stable with larger delays. Scenarios 2, 3 and 4 show similar performances to that of Scenario 1. It can also be seen that, for the same reference queue size, the queue size becomes more and more stable from Scenario 1 to Scenario 4 because of the reasons we have mentioned in Section 4.2.5. PIR-B has the similar behavior as PIR-A, as shown in Figure 4-16.

4.2.7 Effect of Recommended Peak Source Rate μ_{max}

Figure 4-17 shows the average queue size deviations of PIR-A in different scenarios when the recommended peak source rate μ_{max} varies. For Scenario 1, we can see that the system stability is not sensitive to the value change of μ_{max}. The queue size deviation does not show big change when μ_{max} varies from 50 to 100, 200, 300 and 400 packets/sec, though it gets a

little bigger when μ_{max} becomes larger. Scenarios 2, 3 and 4 show similar performances to that of Scenario 1. It can also be seen that, for the same recommended peak source rate, the queue size becomes more and more stable from Scenario 1 to Scenario 4 because of the reasons we have mentioned in Section 4.2.5. PIR-B has the similar behavior as PIR-A, as shown in Figure 4-18, except that the performance of all the scenarios are not much different from each other.

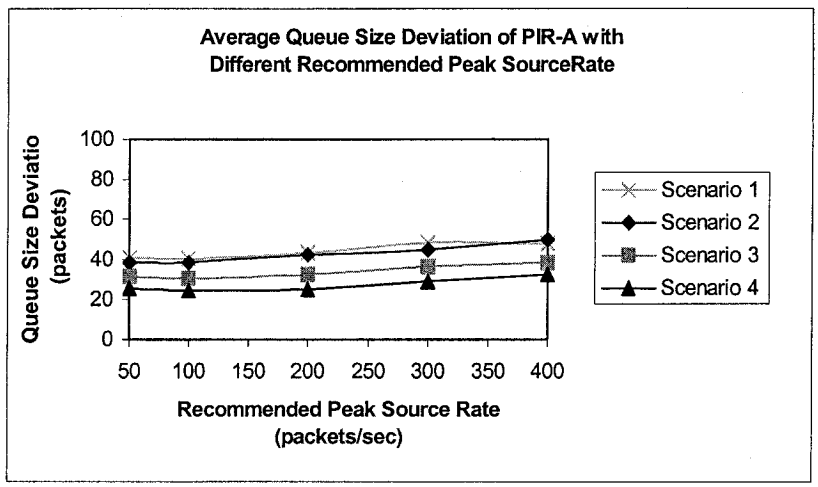


Figure 4-17 Average Queue Size Deviation of PIR-A in Wired Network with Different Recommended Peak Source Rate μ_{max} (q_{SP} = 100, k = 0.08 and d = 0.5)

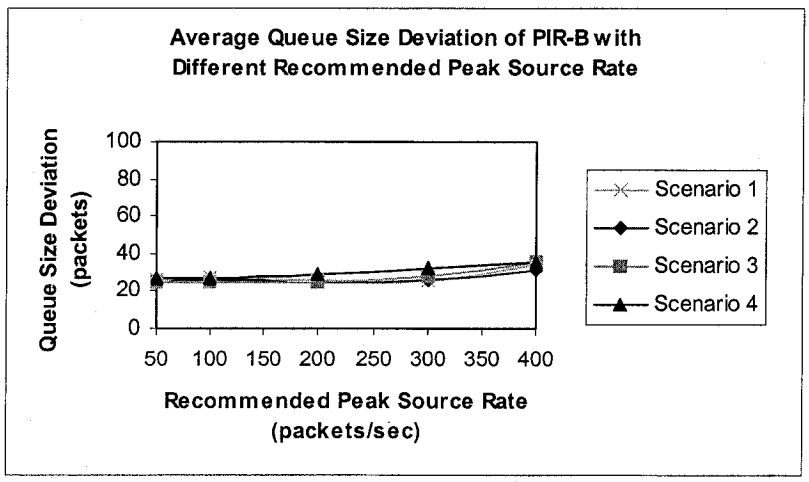


Figure 4-18 Average Queue Size Deviation of PIR-B in Wired Network with Different Recommended Peak Source Rate μ_{max} (q_{SP} = 100, k = 0.08, d=0.5 and Pre-set RTT=0.06 sec)

Chapter 5
Performance Evaluation in Mixed (Wired/Wireless) Networks

As mentioned in Chapter 3, the PIR controller computes a window size according to the real buffer occupancy status and proposes its value to the TCP source. Unlike that of RED-related schemes, which shrink the source window size unnecessarily upon detection of packets loss due to the wireless link problem, one good application of our algorithm is on wireless communications. In this case, our PIR controller does not have to reduce its window size in the event of packet loss due to wireless link errors. In this chapter, we will study via simulations various performance issues of the PIR controllers in the mixed (wired/wireless) network configuration.

5.1 Simulation

We verify our proposition via simulations using the OPNET simulator. The simulation measurements are taken from the bottleneck router (Router 1).

To validate the performance of our PIR controller, we compare its OPNET simulation results in both implementation cases (PIR-A and PIR-B) with those of RED, ECN, and PI-RED. In evaluating the controllers in the mixed network, we use the same four performance measures (instantaneous buffer queue size, packet drop rate, average throughput and the average queue size deviation) that are used to evaluate the PIR controllers in the wired network, except that we measure the throughput in the wireless node instead in the bottleneck router, because the throughput improvement in the wireless node is one of the major concerns in the mixed network.

All network systems are simulated for a time period of 120 seconds, with about 250 packets received in the router per second on the average. In the figures for steady-state performance analysis, each individual point is the average of 4 different simulation runs. This enabled us to have obtained 95% confidence interval for an error criterion ± 1 packet for the average queue size deviation measure.

Table 5-1 Link Error Rates in Different Simulation Scenarios for Mixed Network

Scenario No.	Wireless link error rate
5	0.0025
6	0.01
7	0.04

Different from Scenarios 1 to 4 for wired network, Scenario 5 to Scenario 7 are based on the mixed network, but with different average link error rates (see Table 5-1). We choose to study the case for the link error rate to be 0.01 (Scenario 6) because that error rate is typical for a wireless channel. In order to see the trend of the performance change, we also choose a scenario with a smaller error rate (Scenario 5) and a scenario with a bigger error rate (Scenario 7).

In these scenarios, there are 8 wired sources and 1 wireless source. They are all ftp sources and start to send packets from the beginning of simulation. All the connections have the same propagation delay of 40 ms.

The parameters of RED, ECN, PIR-A and PIR-B are the same as those for the wired network scenarios in Section 4.1. We changed two parameters for PI-RED to reflect the change of round-trip delay and network load. The round-trip time limit R_0 is 0.05, and the load factor N is 9.

5.2 Performance Evaluation

We provide below the simulation results for Scenarios 5, 6 and 7 and our observations.

5.2.1 Scenario 6

Scenario 6 is studied first because it uses a typical link error in the wireless communication environment.

Figures 5-1a to 5-1e and Figures 5-2a to 5-2e show the buffer queue size and the packet drop rate in Router 1. Figure 5-7b compares the average throughput of the wireless node with different controllers.

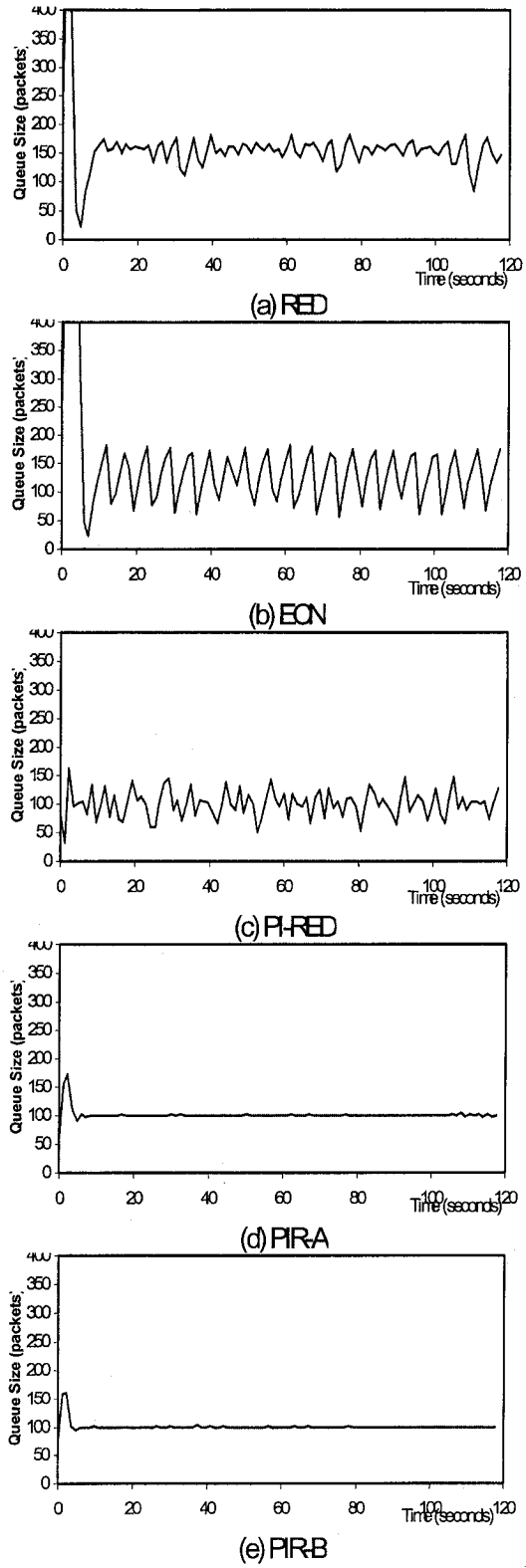


Figure 5-1 Evolution of Queue Size under Different Control for Scenario 6

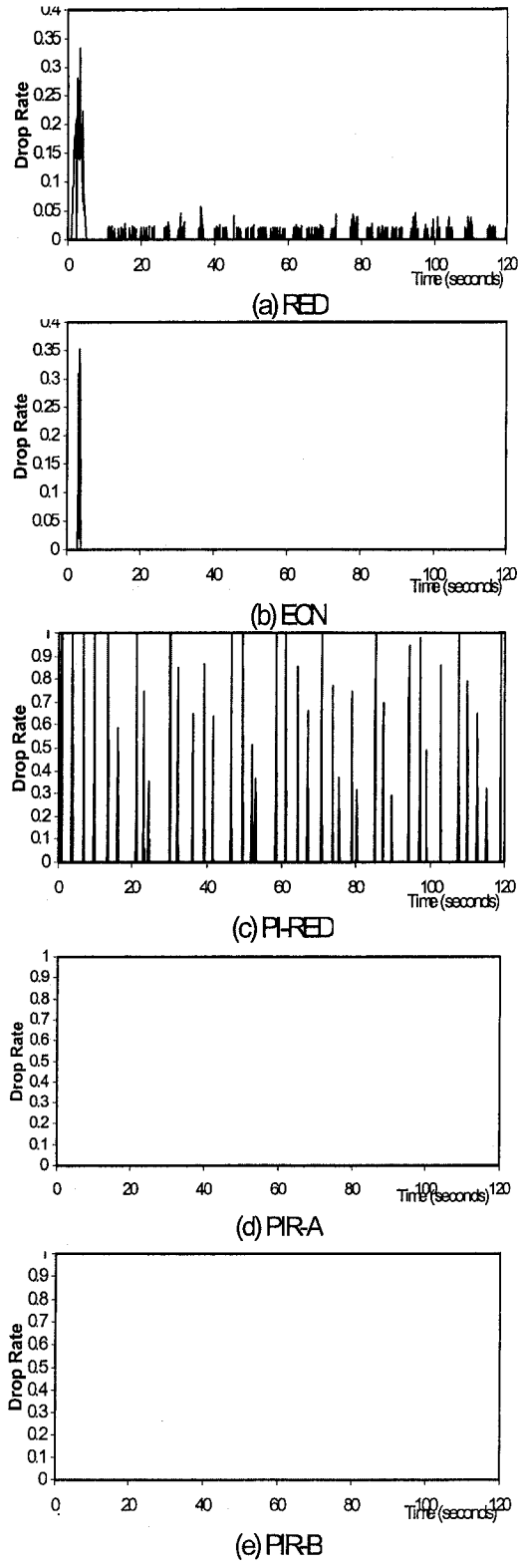


Figure 5-2 Packet Drop Rate under Different Control for Scenario 6

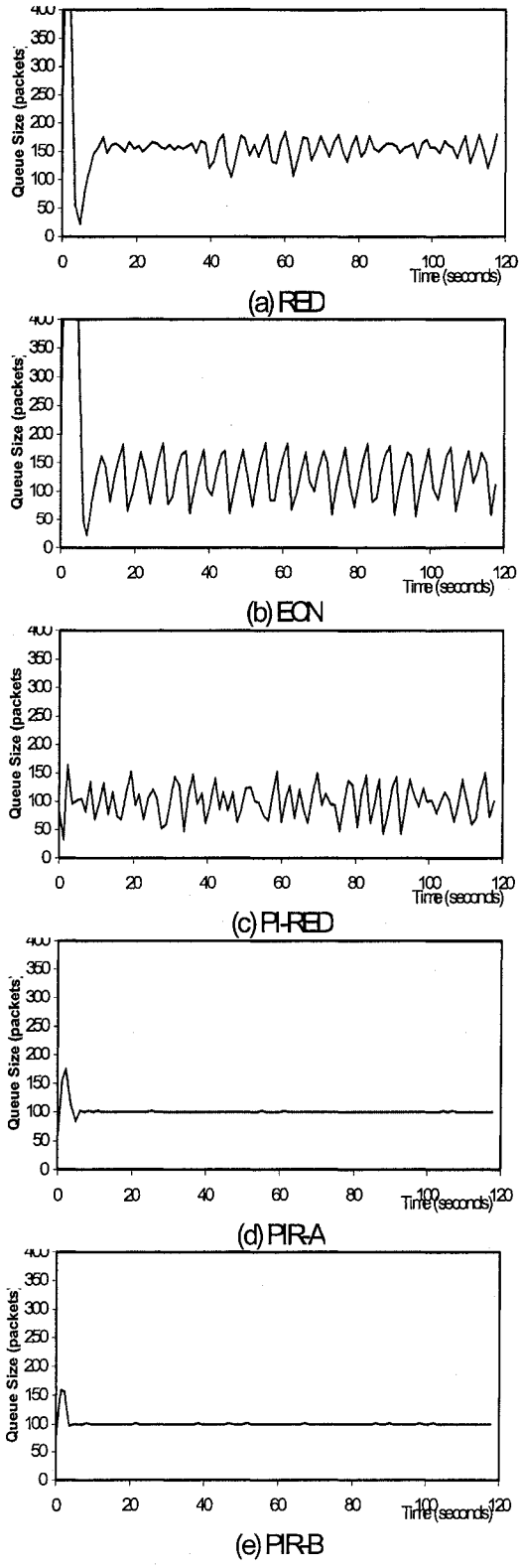


Figure 5-3 Evolution of Queue Size under Different Control for Scenario 5

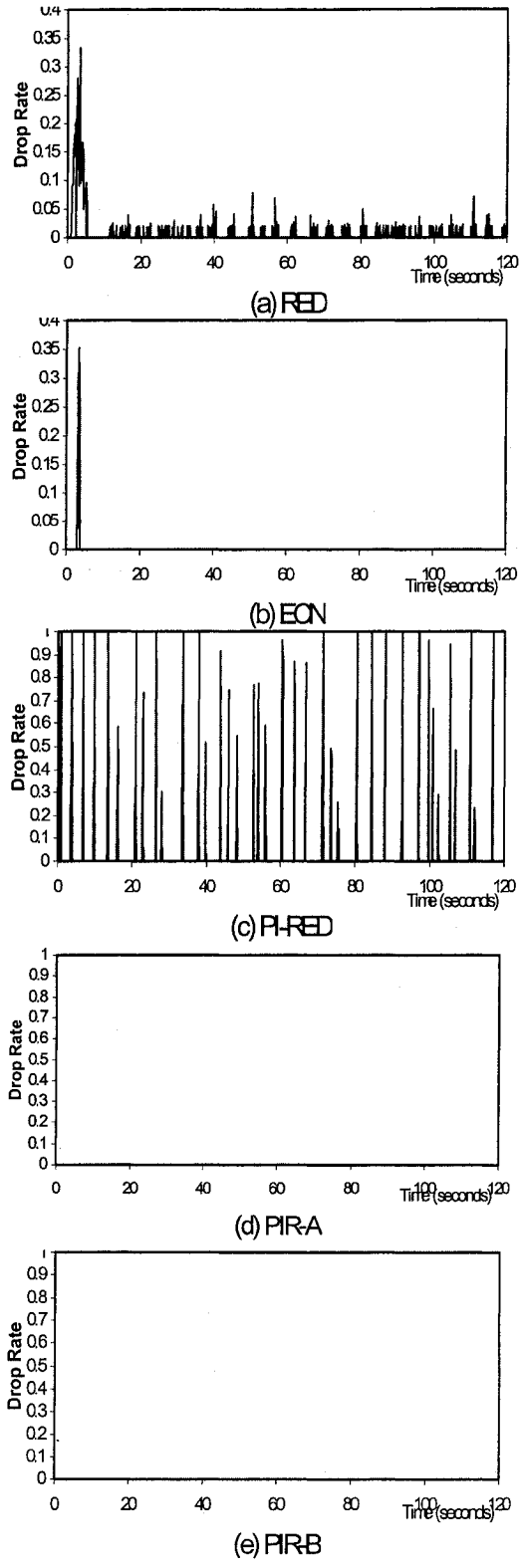


Figure 5-4 Packet Drop Rate under Different Control for Scenario 5

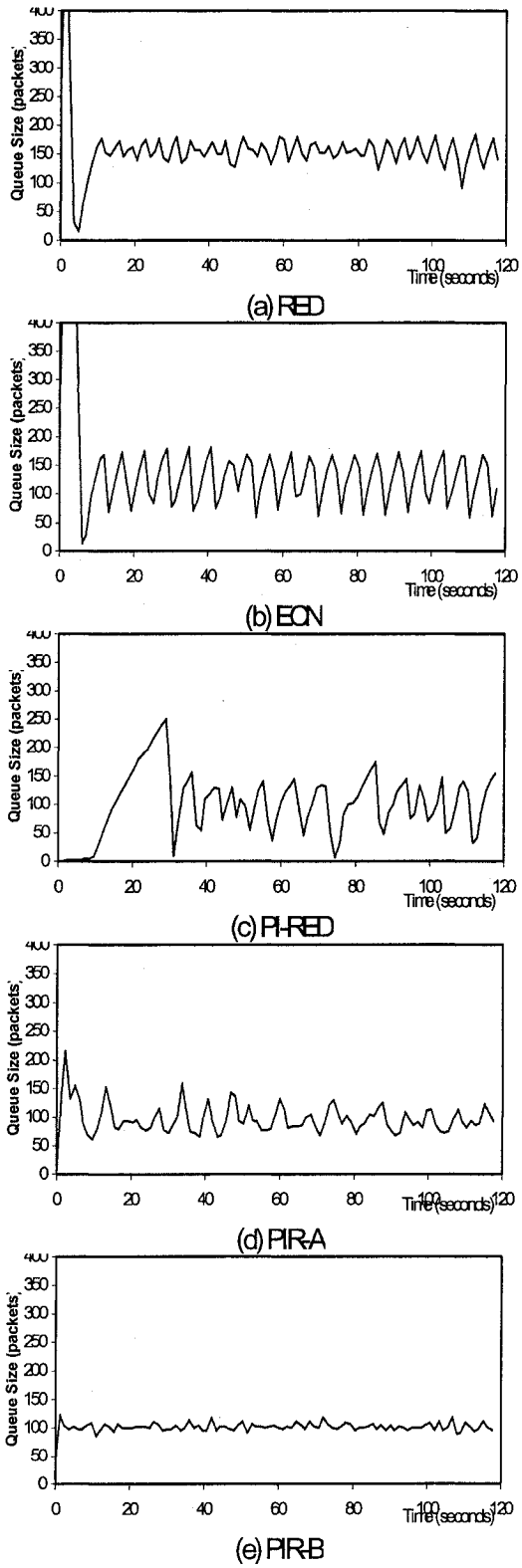


Figure 5-5 Evolution of Queue Size under Different Control for Scenario 7

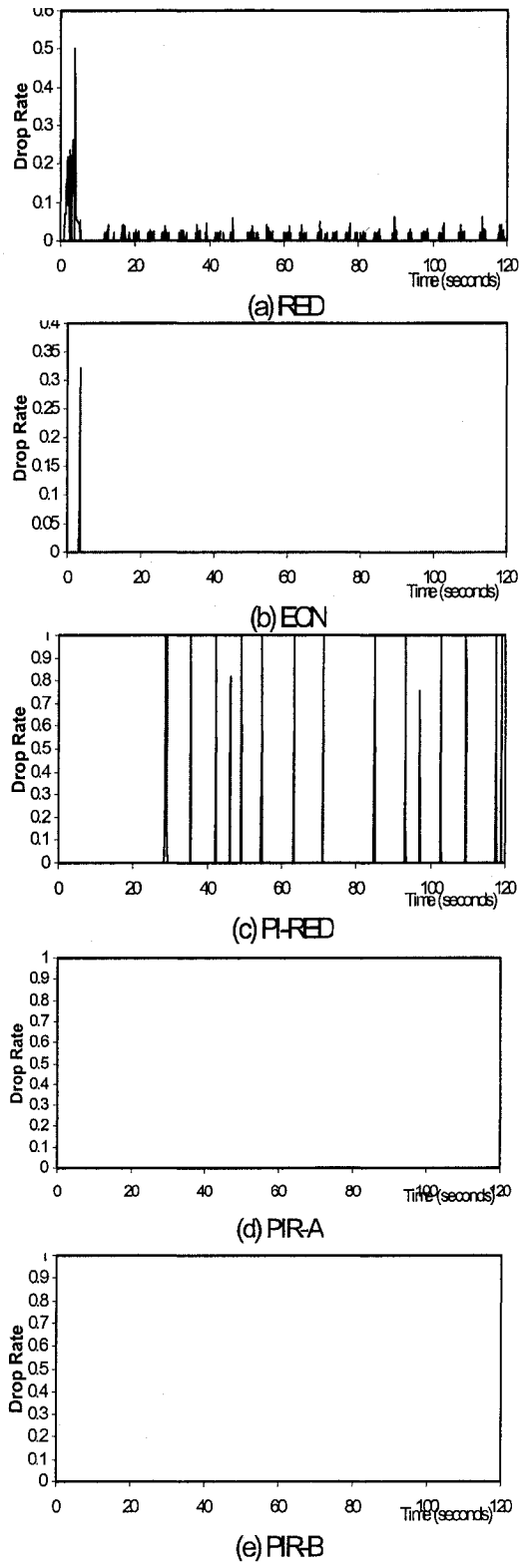


Figure 5-6 Packet Drop Rate under Different Control for Scenario 7

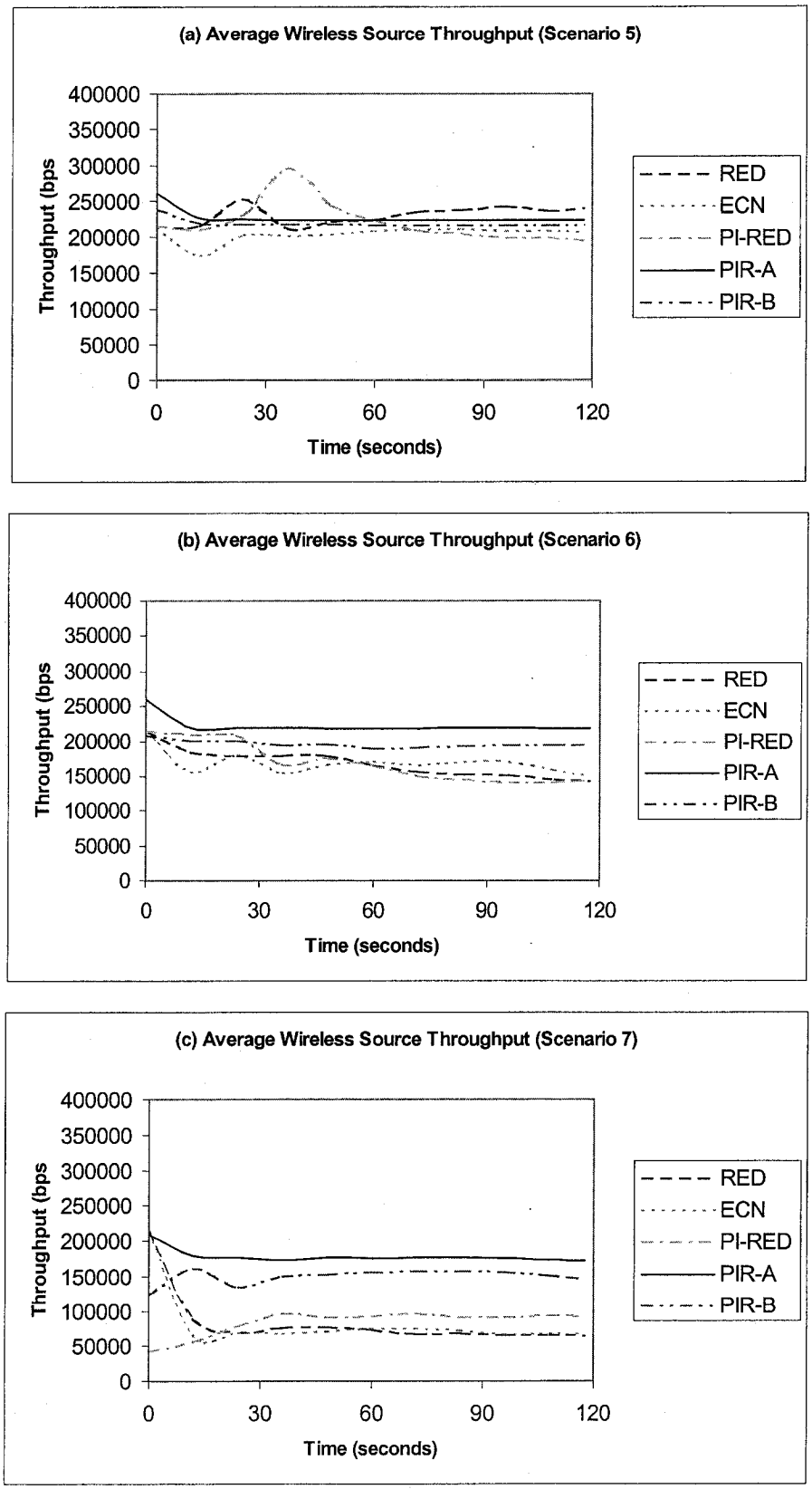


Figure 5-7 Average Throughput of Wireless Node in Mixed Network

From Figure 5-1a, we can see that RED shows obvious queue size oscillation. As shown in Figure 5-1b, the queue size of ECN fluctuates even more widely. Both RED and ECN have a high initial queue size because all the sources send packets aggressively in the beginning of simulation. It can be seen from Figure 5-1c that PI-RED can control the queue size to be around the reference queue size, which is 100. However, the queue size oscillation is still noticeable. We can see from Figure 5-1d to 5-1e, that the PIR controller can keep the router queue size quite stable even in the presence of wireless links.

As can be seen from Figure 5-2a that the RED router always drops packets in order to stay uncongested. The highest packet drop rate appears in the beginning, when a significant part of the packets entering the router are dropped. ECN behaves similarly to RED in the beginning of simulation. As can be seen from Figure 5-2b, many packets are dropped in the start time when packets arrive but the buffer is full. However, in the remaining time, no more packets are dropped because ECN does not drop packets to indicate the congestion status as RED does. From Figure 5-2c, we can see that PI-RED is similar to RED in dropping packets continuously because it also notifies the source of congestion by discarding packets. However, compared with RED, PI-RED drops packets less frequently, but its drop rate is often much higher. That is because PI-RED controls the packet drop according to the output of its PI controller. If this controller is not tuned perfectly well, then the controller output will show oscillation inevitably. In the case of wireless communication environment, which has packet drop due to link errors, we have found it very difficult to tune PI-RED. Like in the wired network, no packets are dropped from the bottleneck router that is controlled by the PIR controller (Figure 5-2d and 5-2e).

Figure 5-7b shows that the average throughput of the wireless node is much better with PIR-A and PIR-B. The other controllers fail to give the wireless node a fair share of the bandwidth (The fair share for each node should be $2Mbps / 9 \cong 0.22Mbps$). They can only give it a little more than the half of the fair share. However, PIR-A can guarantee the wireless node of the fair share of network bandwidth. As we have known, the other controllers all cannot distinguish if a detected packet loss is due to network congestion or wireless link error, so they shrink their window size no matter if the packet loss is caused by traffic congestion or not. This problem can be avoided by PIR-A. The reason that PIR-B cannot achieve the same wireless node throughput performance is because the window size of the

source is also limited by the congestion window size *cwnd* (see Equation (3-39)). Though the advertised window size of PIR-B is not affected by the wireless link error, the congestion window size is decreased unnecessarily. Thus PIR-B performs a little worse, compared with PIR-A. However, it is still much better than RED, ECN and PI-RED. In addition, the wireless node throughput of PIR-A and PIR-B is much more stable than that of the other three controllers.

5.2.2 Scenarios 5 and 7

Figures 5-3a to 5-3e and Figures 5-4a to 5-4e show the buffer queue size and the packet drop rate in Router 1 for Scenario 5. As we can see, the queuing performances (both the queue size and the packet drop rate) in the router in Scenarios 5 are very similar to that in Scenario 6. Hence, we will not discuss the queuing performance here. In stead, we compare the average throughput of the wireless node for each controller.

Figure 5-7a shows that when the link error rate is low (0.0025 for Scenario 5), the wireless node throughput of all controllers does not show big differences. Both PIR-A and PIR-B have a little higher throughput than ECN and PI-RED in most of the time. RED has almost the same throughput as PIR-A and PIR-B, and shows a little better than ECN and PI-RED, because it drops packets more frequently and cause fast retransmission of packets more frequently. Hence, it has more packets sent out than ECN and PI-RED. It can be obviously seen that the wireless node throughput of PIR-A and PIR-B is much more stable than that of the other three controllers. Besides, PIR-A and PIR-B do not drop packets.

Figures 5-5a to 5-5e and Figures 5-6a to 5-6e show the buffer queue size and the packet drop rate in Router 1 for Scenario 7. Figure 5-7c compares the average throughput of the wireless node with different controllers for Scenario 7.

As we can see from Figure 5-5a and 5-5b, both the queue size and the packet drop rate performances of RED and ECN are very similar to those in Scenarios 5 and 6. RED shows obvious queue size oscillation and the queue size of ECN fluctuates even more widely. It can be seen from Figure 5-5c that PI-RED shows severe queue size oscillation in Scenario 7 because the packets dropped due to wireless link errors are far more than those in Scenarios 5 and 6. As a result, PI-RED has failed in controlling the queue size stability effectively. In the aspect of packet drop rate, PI-RED shows a less frequent packet drop actions, but for each

period of packet dropping, the drop rate is much higher than in Scenarios 5 and 6 due to the same reason mentioned above.

The queue size of PIR-A also shows some oscillation in Scenario 7 because it becomes more and more difficult to control the queue size stability as the link-error-caused packet drops become more frequent. It is the same case for PIR-B. However, PIR-B performs much better than PIR-A in Scenario 7 in terms of queue size stability. The reason is that PIR-B has a shorter time delay in delivering the information of congestion from the router to the source (Ref: Section 3.3). Like in Scenarios 5 and 6, PIR-A and PIR-B still do not drop packets.

As we can see from Figure 5-7c, when the link error rate is large (0.04 for Scenario 7), the wireless node throughput of all controllers decreases, including PIR-A and PIR-B. RED and ECN suffer severely from the high wireless link error because of their incapability of telling the difference between congestion-caused packet drops and link-error-caused packet drops. PI-RED has a little higher throughput than RED and ECN because it does not shrink its window size as much as RED and ECN do due to its lower packet drop rate. In addition, the throughput of all controllers becomes unstable, especially in the beginning. However, PIR-A and PIR-B still outperform the other three controllers significantly.

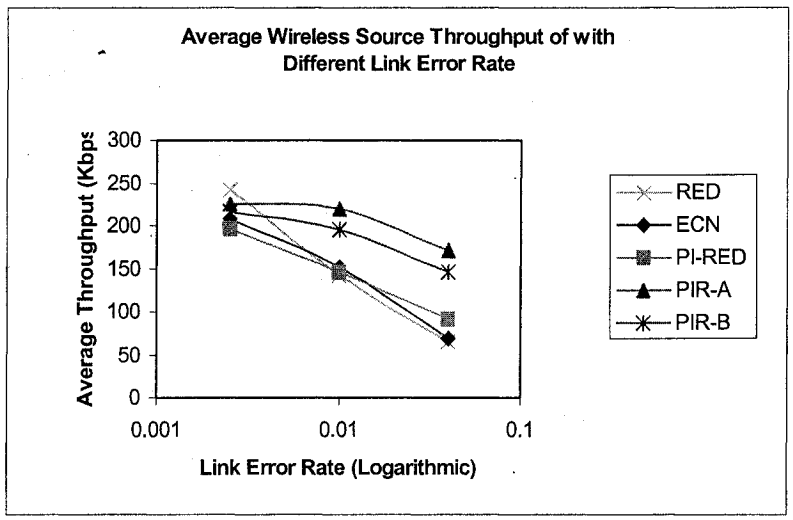


Figure 5-8 Average Wireless Source Throughput of with Different Link Error Rate

5.2.3 Comparison of wireless node throughput for Scenarios 5, 6 and 7

Figure 5-8 shows that when the link error rate is low (0.0025 for Scenario 5), the wireless node throughput of all controllers does not show big differences. Both PIR-A and PIR-B have a little higher throughput than ECN and PI-RED. RED shows the best because it drops packets more frequently and cause fast retransmission of packets more frequently. Hence, it has more packets sent out than other controllers. When the link error rate is 0.01 (for Scenario 6), the average throughput of the wireless node is much better with PIR-A and PIR-B, though PIR-B cannot achieve the same wireless node throughput performance as PIR-A, which is explained in Section 5.2.1 in details. RED, ECN and PI-RED have almost the same throughput performance in this case. When the link error rate is large (0.04 for Scenario 7), the wireless node throughput of all controllers decreases, including PIR-A and PIR-B. However, PIR-A and PIR-B still outperform the other three controllers significantly. Considering all the three scenarios, we can see that RED suffers the most severely when the link error rate becomes large.

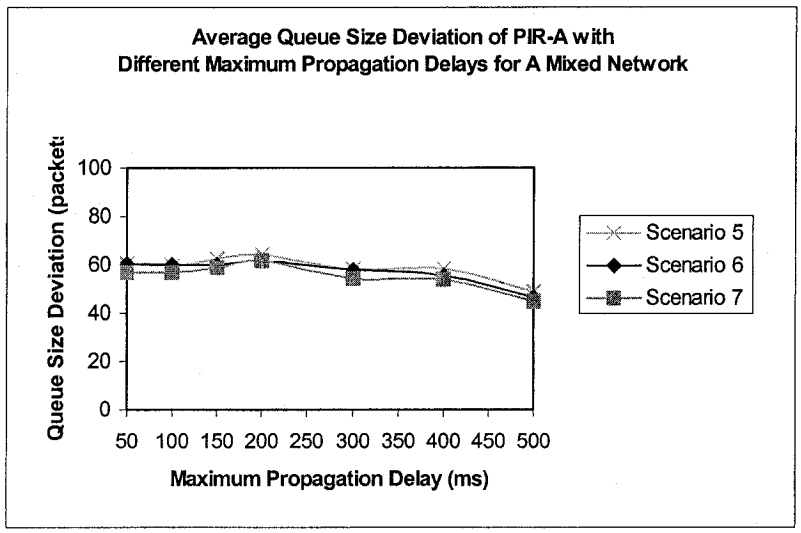


Figure 5-9 Average Queue Size Deviation of PIR-A in Mixed Network with Different Maximum Propagation Delays ($q_{SP} = 100$, $\mu_{max} = 80$, $k = 0.08$ and $d = 0.5$)

5.2.4 Effect of RTT

Figure 5-9 shows the performance of average queue size deviations of PIR-A in a mixed network for different maximum propagation delays. For Scenario 5, we can see that when the maximum propagation delay increases, the average queue size deviation decreases slightly.

Scenarios 6 and 7 have almost the same behavior. In general, when the maximum propagation delay becomes larger, the queue size deviation tends to become smaller. This is quite different from the wired network, where the deviation is increasing. The reason may be that the current controller parameter setting is not the best choice for a mixed network with a maximum propagation delay of 40 ms (Though it can also stabilize the queue size well as seen in Figure 5-1d, 5-1e, 5-3d, 5-3e, 5-5d, and 5-5e). Instead, this controller parameter setting is more suitable for the mixed network with larger maximum propagation delays. The more inferior performance is probably due to the fact we have not found suitable parameters for the mixed network, so the queue size deviation here is even not better than that with higher delays. PIR-B has almost the same behavior as PIR-A, and therefore not shown here.

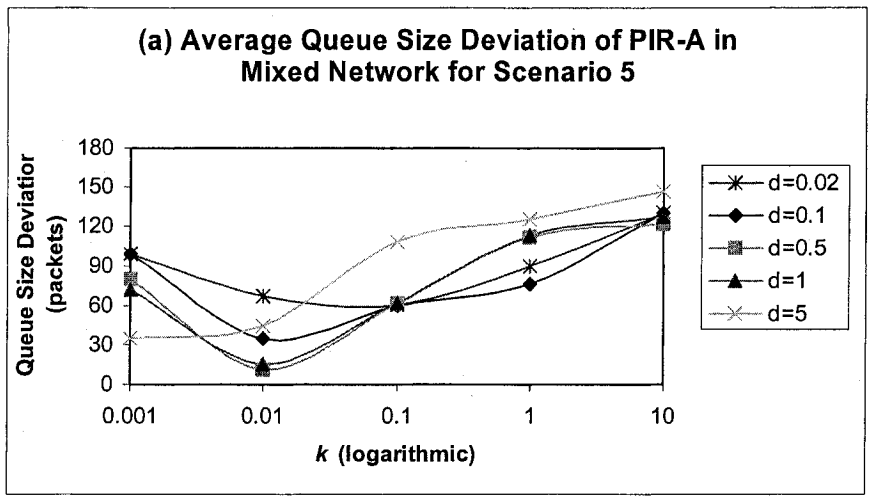
5.2.5 Effect of Proportional Gain k and Adaptation Constant d

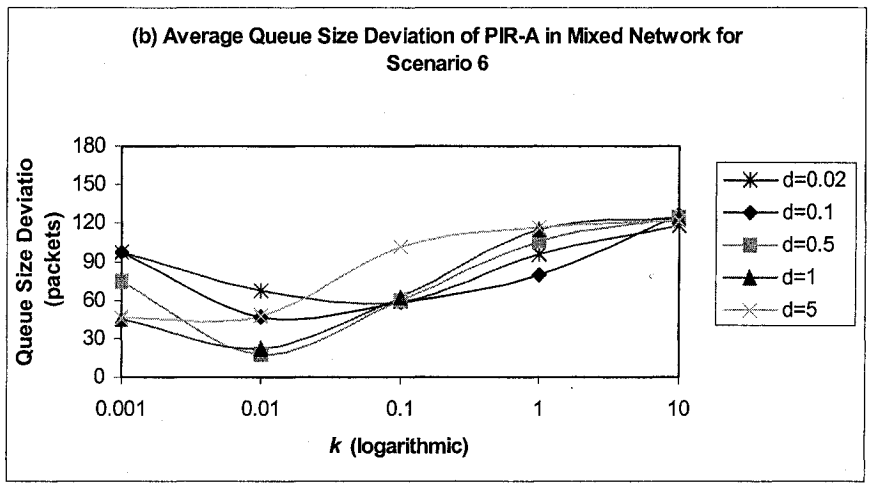
Figure 5-10a shows the average queue size deviation as a function of k for different d values under Scenario 5. In the case of d=0.02, the deviation exhibits a "valley" characteristic with high around k = 0.01 and k = 10 and a low at k = 0.1. Similar observations are obtained for d=0.1, 0.5, 1 and 5, but the "valley" value appears to shift from k = 0.1 to k = 0.01 as d increases from d = 0.02 to d =1. Figures 5-10b and 5-10c show almost the same performances as Figure 5-10a for Scenarios 6 and 7. The reason for the "valley" phenomenon has been examined in last chapter. PIR-B has the similar behavior as PIR-A.

By comparing the performance in the mixed network with that in the wired network, we can find they are very similar to each other, except that the queue size deviation in the wired network is larger than that in the mixed network because all the TCP sources in the mixed network are ftp sources, which all start sending packets in the beginning and stop at the end of simulation, and there is no other sources entering the network at any time. Hence, the queue size is more stable in the mixed network.

5.2.6 Effect of Reference Queue Size q_{SP}

Figure 5-11 shows the average queue size deviations of PIR-A in different scenarios when the reference queue size q_{SP} varies. We can see, for Scenario 5, when q_{SP} is set to be 100,





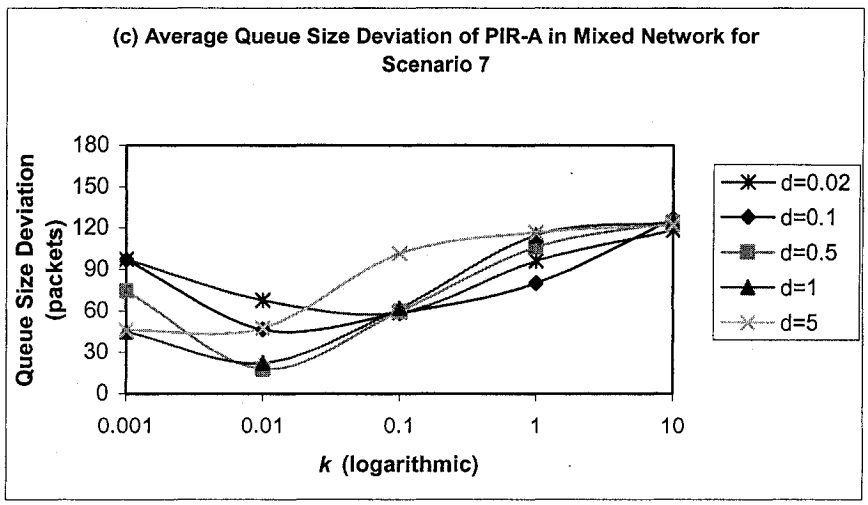


Figure 5-10 Average Queue Size Deviation under Different Combinations of PIR Controller Proportional Gain k and Adaptation Constant d in Mixed Network ($q_{SP} = 100$, $\mu_{max} = 50$)

the smallest queue size deviation is observed. When q_{SP} varies from 200 to 300, 400 and 500, we can see that the queue size becomes less and less stable, though there is a very small fluctuation after q_{SP} is greater than 300. This is because larger reference queue size causes larger queuing delay, which results in a larger round-trip delay. Thus, the system becomes

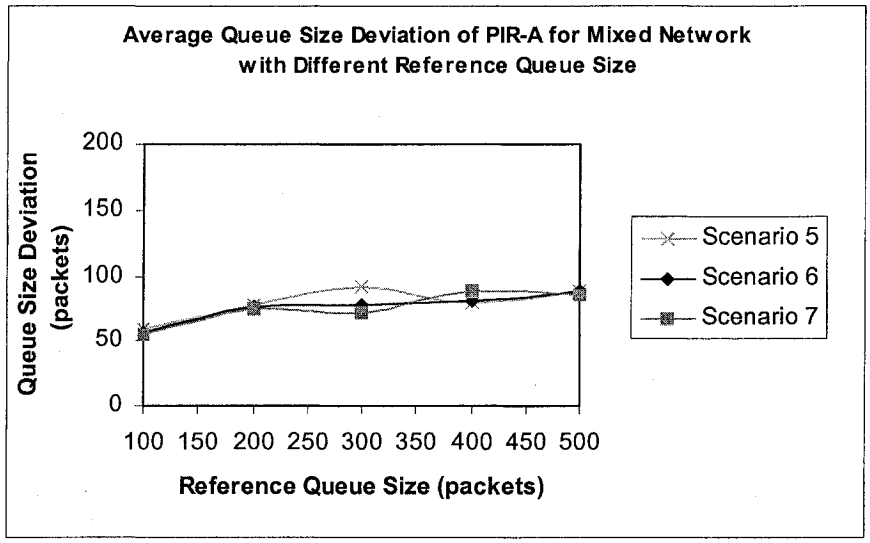


Figure 5-11 Average Queue Size Deviation of PIR-A in Mixed Network with Different Reference Queue Size q_{SP} ($\mu_{max} = 50$, $k = 0.08$ and $d = 0.5$)

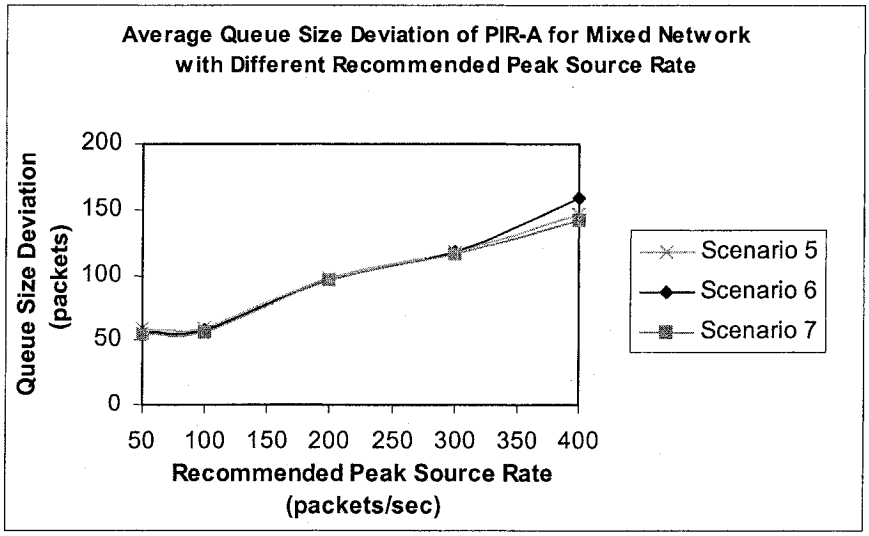


Figure 5-12 Average Queue Size Deviation of PIR-A in Mixed Network with Different Recommended Peak Source Rate μ_{max} ($q_{SP} = 100$, $k = 0.08$ and $d = 0.5$)

less stable with larger delays. Scenarios 6 and 7 show similar performances to that of Scenario 5. PIR-B has the similar behavior as PIR-A.

5.2.7 Effect of Recommended Peak Source Rate μ_{max}

Figure 5-12 shows the average queue size deviations of PIR-A in different scenarios when the recommended peak source rate μ_{max} is set to 50, 100, 200, 300 and 400 packets/sec, respectively. For Scenario 5, we can see that the queue size deviation increases almost linearly with respect to the recommended peak source rate. Scenarios 6 and 7 show similar performances. This is because a larger recommended peak source rate produces similar effect on the source rate to that produced by a larger reference queue size. This can be inspected from Equation (3-4) in that $\mu(t)$ is an increasing function of μ_{max} or q_{SP}. Hence, the larger reference queue size causes larger queue size deviation as explained in Section 5.2.6. PIR-B has the similar behavior as PIR-A.

Chapter 6
Design Issues and Guideline

In this chapter, we will give a design guideline for our PIR controller, specifically, PIR-A and PIR-B, based on the stability analysis in Chapter 3 and our simulation experience in Chapters 4 and 5. We will discuss how to choose the various controller parameters in order to have a best-designed PIR controller in terms of the performance measures we have examined in the last two chapters, e.g. the queue size stability and the packet drop rate.

6.1 PIR-A

As we have seen in the last two chapters, choosing different parameters for PIR-A can have different queue size stability in the bottleneck router. How to find the best parameters is the key factor for PIR-A controller design. There are totally four different controller parameters for PIR-A. They are the reference queue size q_{SP}, the recommended peak source rate $\mu^{\max}$, the proportional gain k, and the adaptation constant d.

1. *Reference queue size* q_{SP}: Basically, a larger reference queue size will bring a less stable buffer queue. In addition, it will introduce a larger queuing delay. So, unless necessary, a large reference queue size shall be avoided. We recommend a reference queue size value close to 100.
2. *Recommended peak source rate* $\mu_{\max}$: This rate can be computed roughly as the bottleneck router service capacity (in packets) divided by the number of active connections on the average. In the case that the exact computation is not feasible, an initial value set from the rough estimation can also be used to achieve good performance.
3. *Proportional gain* k *and adaptation constant* d: These two parameters play the dominant roles in the designed controller quality. The choosing of one of these two parameters will affect the range of the acceptable values to be chosen for the other parameter. Three key factors will determine how these two parameters can be chosen. They are the service capacity of the bottleneck router, the maximum round-trip delay and

the maximum number of flows passing through the bottleneck router. In choosing these two parameters, we can refer to the controller stability criteria derived in Chapter 3 and shown as follows.

$$0 < k < \frac{1}{2N\tau_{\max}}, \qquad 0 < d \text{ and } \left(\frac{sd}{\beta Nk} + 2 + \frac{2\beta Nk}{sd}\right) \cdot \frac{\sqrt{2} + \frac{sd}{\beta Nk}}{\frac{1}{\sqrt{2}\beta Nk\tau_{\max}} - 1} - 1 < 0 .$$

However, the selection process should not be limited by the above criteria, because the criteria we derived in Chapter 3 are the conservative criteria. The values out of that range may also be accepted to achieve a stable buffer queue size in the bottleneck router. Besides, sometimes we can choose to keep a tiny queue size oscillation in exchange of the small instant queue size deviation when the traffic load varies greatly (In actual control application, a quick recovery to the system stabilization often accompanies a short period of large system oscillation). We prefer to choose a k value a little larger than the values within the stability criterion region to allow a tiny queue size oscillation, and a d value a little smaller than the values within the stability criterion region to avoid the short period of large system oscillation. In our simulated network environment, we choose to let k be between 0.01 and 0.1 and d between 0.1 and 1. For example, we used a wide range of the values for k and d to show the region of k and d which can perform best in Figure 4-14. The combination of $k = 0.01$ and $d = 1$ is a proper setting for Scenario 1, whereas, the combination of $k = 0.001$ and $d = 1$ is not.

All the parameters need to be set for PIR-A have been examined above. As for the other factors, e.g. the time delay and the maximum number of traffic flows, they are not the parameters for the controller and are beyond the control of the user. The user can then adjust the above parameters to adapt to the environmental factors.

6.2 PIR-B

The design guideline and recommended values for PIR-A can also be used to design PIR-B. The only other issue needed to be concerned in the PIR-B controller design is how to choose the pre-set fixed round-trip time *RTT*, because PIR-B used the fixed *RTT* instead of the real

round-trip delay measured by the TCP source in the computation of the advertised window size in the router.

We recommend a pre-set *RTT* in the range of 0~0.3 second even though this may be different from real measured RTT. This is because the queue size stability in the bottleneck router dose not appear to be affected by the choice of the pre-set *RTT* (see Figure 4-10 for example).

6.3 Multiple-Router Implementation

In a multiple-router environment, we suggest a method to implement the PIR controller for either PIR-A or PIR-B as follows. Each time when a router sends out the received packet, it checks the value of the advertised window size that is already in the packet (in the case of PIR-A) or ACK (in the case of PIR-B) header. If this value is smaller than or equal to the advertised window size value computed in the current router, the router then does not change the value. Otherwise, the value in the packet/ACK header will be replaced by the one computed in the current router.

Chapter 7
Conclusions

We have proposed a Proportional and Integral rate controller to regulate TCP traffic in the network. This controller is based on the classical feedback control theory. We introduced the integral part into the proportional rate controller to eliminate the steady state error problem inherent in the pure proportional control. We proved the PIR controller can stabilize the network and gave the stability criteria. We also proposed two different implementation methods for the PIR controller, i.e. PIR-A and PIR-B. As we have pointed out, PIR-A, like ECN, requires the support from both router and end node, so it is aimed for future applications, where high traffic stabilization is requested. Currently, PIR-B is suitable for edge network areas, as it only requires the support from router.

In evaluating the performance of our PIR controller, we conducted a large number of simulations in two different network environments, the wired network and the mixed (wired/wireless). First in the wired network, we compared both of PIR-A and PIR-B with RED, ECN and PI-RED in four performance aspects in the bottleneck router, the instantaneous buffer queue size, the packet drop rate, the average router throughput and the average queue size deviation. We have seen our PIR controller outperforms the other three controllers in all the cases. It can achieve the most stable instantaneous buffer queue size, the lowest packet drop rate, the highest average router throughput and the smallest average queue size deviation. We also studied the effects of various controller parameters of the PIR controller on the queue size stability in the bottleneck router, which can help to design our PIR controller properly.

Next, we compared PIR-A and PIR-B with RED, ECN and PI-RED in the mixed (wired/wireless) network in the same four aspects as in the wired network, except that we compared the average throughput of the source instead of that of the router. Like with the wired network, we found our PIR controller outperforms the other three controllers in all the cases. Especially, It can greatly improve the data throughput of the wireless source in a typical wireless communication environment.

Based on the theoretical stability analysis and the simulation observations of our PIR controller, we provided the detailed guideline to designing PIR-A and PIR-B. In addition, we also suggested a design method to implement the PIR controller in a network with more than one bottleneck routers.

7.1 Future Work

There are still some remaining issues to be investigated regarding the implementation of the PIR controller in a multiple-router network. In addition, we are also interested in finding how our PIR controller performs with the concern of QoS issues, e.g. different types of services and packet priorities. Our preliminary studies have indicated that the high queue size stability and throughput, combined with the other advantages, like fairness and low packet loss rate, makes the PIR controller a powerful controller for the TCP traffic and congestion control.

References

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

[AwOu01] J. Aweya, M. Ouellette, D.Y. Montuno and A. Chapman, "A Control Theoretic Approach to Active Queue Management", *Computer Networks*, Vol. 36, pp. 203-235, 2001.

[AwOu02] J. Aweya, M. Ouellette and D.Y. Montuno, "TCP Rate Control with Dynamic Buffer Sharing", *Computer Communications*, Vol. 25, pp. 922-943, 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, Dec. 1997.

[BaSe95] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, "Improving TCP/IP Performances over Wireless Networks", *Proceedings of IEEE Mobicom 95.* pp. 2-11, Berkeley, California, November 1995.

[BeMe93] L. Benmohamed and S. M. Meekov, "Feedback Control of Congestion in Packet Switching Networks: The Case of A Single Congested Node", *IEEE/ACM Transactions on Networking,* vol. 1, pp. 693-707, 1993.

[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*, pages 24--35, London, UK, May 1994.

[CaGe00] C. Casetti, M. Gerla, S. S. Lee, S. Mascolo and M. Sanadidi, "TCP with Faster Recovery", *Proceedings of IEEE Milcom 2000*, pp. 320-324, Los Angeles, CA, October 2000.

[CaGe01] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", *Proceedings of IEEE Mibicom 2001.* pp 287-297, Rome, Italy, July 2001.

[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, Dec. 1997.

[FaFl96] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communication Review*, V. 26, N. 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.

[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.

[FlFa99] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking,* vol. 4, pp. 458-472, Aug. 1999.

[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.

[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.

[Floy91] S. Floyd, "Connection with Multiple Congested Gateways in Packet Switched Networks Part 1: One-way Traffic", *ACM Computer Communication Review,* V. 21, N. 5, Oct. 1991.

[Floy94] S. Floyd, "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, V. 24 N. 5, p. 10-23, October 1994.

[GeSa01] M. Gerla, M. Sanadidi, R. Wang and A. Zanella, "TCP Westwood: Congestion Window Control Using Bandwidth Estimation", *Proceedings of IEEE Globecom 2001.* San Antonio, Texas, USA, November 2001.

[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, Nov. 2000.

[Hale97] J.K. Hale, *Theory of Fundamental Differential Equations,* Springer-Verlag, New York, 1997.

[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, Apr. 2001.

[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, Apr. 1990. ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt

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

[Khal96] H. Khalil, *Nonlinear Systems*. Prentice Hall, New Jersey, 1996.

[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.

[LaKe00] K. Laevens, P. B. Key and D. McAuley , "An ECN-based End-to-End Congestion-Control Framework: Experiments and Evaluation", *MSR Technical Report*, MSR-TR-2000-104, October 2000 http://research.microsoft.com/research/network/publications/MSRTR2000_104.pdf

[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.

[LiMo97] D. Lin and R. Morris. "Dynamics of Random Early Detection". *Proceedings of Sigcomm 1997*. pp. 127-137, Cannes, France, September 1997.

[LoPa02] S. H. Low, F. Paganini, J. Wang, S. Adlakha and J. C. Doyle, "Dynamics of TCP/RED and a Scalable Control", *Proceedings of IEEE Infocom 2002*. New York, 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", October 1996, *IETF RFC2018.* 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, Dec. 1999.

[MaSe97] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoid Algorithm", *ACM Computer Communication Review,* V. 27, N. 3, pp. 67-82, Jul. 1997.

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

[Morr00] R. Morris, "Scalable TCP Congestion Control", in *Proceedings of IEEE Infocom 2000*. pp. 205--211, October 1997, Atlanta, GE, USA, Oct. 1997.

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

[Ogat90] K. Ogata, *Modern Control Engineering*, Prentice Hall, New Jersey, 1990.

[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.

[PaFl95] V. Paxson and S. Floyd, "Wide-area traffic: The failure of Poisson modeling", *IEEE/ACM Trans. on Networking*, vol. 3, No. 3, pp. 226--244, 1995.

[RaJa90] K.K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks", *ACM Transactions on Computer Systems,* Vol. 8 No. 2, pp. 158-181, 1990.

[SaFe99] D. Saha, W. Feng, D. Kandlur and K. Shin, "BLUE: A New Class of Active Queue Management Algorithms", *Technical report CSE-TR-387-99, University of Michigan*, Aptil 1999.

[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, Apr 1975.

[ShPa97] S. Shenker, C. Partridge and R. Guerin, "Specification of Guaranteed Quality of Service", *RFC 2212*, Sep 1997. http://www.faqs.org/rfcs/rfc2212.html.

[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*, Jan. 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.

[WiDe01] J. Widmer, R. Denda and M. Mauve, "A Survey on TCP-Friendly Congestion Control", *Special Issue of the IEEE Network Magazine "Control of Best Effort Traffic"*, vol. 15, no. 3, pp. 28--37, May/June 2001.

[ZhCo03] H. Zhang, J. Cong and O. Yang, "Rate Control over RED with Data Loss and Varying Delays", *Proceedings of Globecom 2003*, San Francisco, Dec 2003, Session NG11-4.

[Zhu02] Chenyu Zhu, "Analysis and Performance Evaluation of Congestion Control and Avoidance Algorithm in TCP/IP", *Master Thesis, University of Ottawa*, 2002.

[ZhWu02] M. Zhang, J. Wu, C. Lin and K. Xu, "Rethink the Tradeoff between Proportional Controller and PI Controller", ", *Proceedings of IEEE ISCC'02*. pp.57-62, Taormina-Giardini Naxos, Italy. July, 2002.

[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, Apr. 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, Jun. 2000.

Appendix A1
Lyapunov Second Stability Criteria

Lyapunov Second Stability Criteria [Ogat90] is a method used to prove the stability region for a control system in modern control theory. We summarize it as follows.

Suppose that a system is described by

$$\dot{x} = f(x,t), \tag{A1-1}$$

where x is a state vector (n-vector), $f(x,t)$ is an n-vector whose elements are functions of $x_1, x_{2,\cdots,} x_n$, and t, and

$$f(0,t) = 0 \quad \text{for all } t. \tag{A1-2}$$

If there exists a scalar function $V(x,t)$ having continuous, first partial derivatives and satisfying the following conditions:

1. $V(x,t)$ is positive definite
2. $\dot{V}(x,t)$ is negative definite

then the equilibrium state at the origin is uniformly asymptotically stable.

If, in addition, $V(x,t) \to \infty$ as $\|x\| \to \infty$, then the equilibrium state at the origin is uniformly asymptotically stable in the large.

Appendix A2
RED

RED (Random Early Detection) was originally proposed [FlJa93] to overcome the severe performance problems with the earlier gateway queue management algorithm, e.g. the bias against bursty traffic and global synchronization [FlJa93]. The RED gateway uses a low-pass filter with an exponential weighted moving average to calculate the average queue size avg, which is then compared to 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 greater than $th_{\max}$, then every 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.

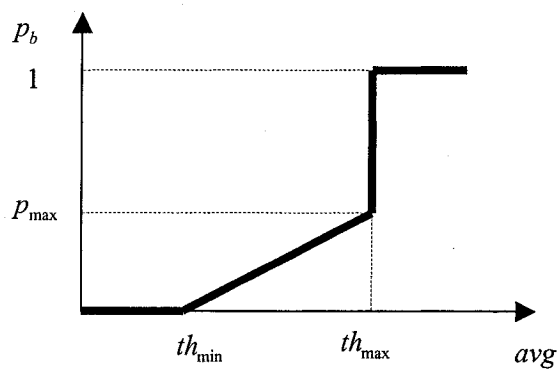


Figure A2-1 Temporary Dropping Probability of RED

Equation (A2-1) shows how the temporary packet-dropping probability p_b is calculated, which is also indicated in Figure A2-1. The final probability p_a with which an arriving packet is dropped is calculated in Equation (A2-2), from which we can see that RED ensures that the gateway does not wait too long before dropping a packet.

$$p_b = p_{\max}(avg - th_{\min})/(th_{\max} - th_{\min}) \quad , \qquad \text{(A2-1)}$$

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 $th_{\min} \le avg < th_{\max}$

$count++$

calculate p_a

$p_b := p_{\max}(avg - th_{\min})/(th_{\max} - th_{\min})$

$p_a := p_b/(1 - count \cdot p_b)$

drop the incoming packet with p_a

$count := 0$

else if $th_{\max} \le avg$

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

$f(t)$: ali near function of the time t

Figure A2-2 RED Gateway Algorithms

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

$$p_a = p_b / (1 - count \cdot p_b) \quad , \tag{A2-2}$$

where *count* is the number of packets since last dropped packet.

Figure A2-2 shows the RED algorithm, which consists of two parts. The first part is to determine the degree of burstiness allowed in the gateway queue through the computation of the average queue size. The second determines how frequently the incoming packets should be dropped according the current congestion level.

Appendix A3
ECN

Generally, the ECN (Explicit Congestion Notification) algorithm that people talk about means the ECN mechanism that was proposed by Sally Floyd [Floy94]. In the fact, there are some other ECN mechanisms before the one proposed in [Floy94], such as Source Quench messages and DECbit's ECN bit. A router or host might send an ICMP (Internet Control Message Protocol) Source Quench message when it receives datagrams at a rate that is too fast [Stev94]. Though RFC 1009 required Internet routers to use Source Quench message mechanism, it was later criticized as consuming network bandwidth and being ineffective and unfair. The DECbit congestion avoidance scheme [RaJa90] uses a congestion notification bit in packet headers to provide feedback information about congestion to the data sender. The sender decreases its congestion window size multiplicatively if more than half of the ACKs for the packets in the last window have the ECN bit set. In contrast to Source Quench messages and DECbit's ECN bit, the ECN scheme in [Floy94] (called ECN in this thesis) has a different mechanism which mainly aimed at solving the unnecessary packet drop problem in RED.

The basic mechanism of ECN is that the TCP source tells the router that the end nodes are ECN-capable by setting the ECN-capable Transport (ECT) bit in its packet IP header. The router sets the Congestion Experienced (CE) bit in the IP header of incoming packets if congestion is detected, and passes the packets to the destination. In turn, the destination will copy the CE bit to the IP header of the ACKs and send them back to the source. The TCP source then reduces its transmission rate in response to the ACK with the CE bit set. It can be seen that the operation mechanism of ECN has two parts. One is in the router. The other is in the end node. The ECN router makes use of the packet marking option provided by RED. It has a low-pass filter to calculate the average queue size, which is then compared to two preset parameters, the minimum threshold and the maximum threshold, to determine if the arriving packets will be marked with congestion by setting the ECN bit in their IP headers. The ECN router algorithm is the same as the RED gateway algorithm shown in Figure A2-2, except that ECN takes the option of marking, instead of dropping. While the TCP sources

work as black boxes to RED, ECN indeed needs TCP to response properly to the feedback of congestion in the network.

TCP's response to ECN was suggested as follows [Floy94]. TCP's response to ECN should be similar to its response to a dropped packet as in indication of congestion. 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 congestion window size and slow start threshold in the case of Reno. TCP should ignore succeeding ECNs if it has reacted to a previous ECN or a dropped packet until all the outstanding packets have been acked. The response to an ECN does not trigger the sending of any new or retransmitted packets. Figure A3-1 shows part of the Reno-ECN algorithm.

```
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 A3-1 Part of Reno-ECN Algorithm

Appendix A4
PI-RED

Since RED has a serious queue size oscillation problem, many researchers have developed RED variants to stabilize the router queue size and maintain it at a desired level. C.V. Hollot et al. proposed a PI (Proportional and Integral) controller to perform in the RED capable router [HoMi01b]. They pointed out two limitations of RED. The first limitation is the tradeoff between response speed and queue size stability. A fast response design will exhibit relatively low stability margins. On the other hand, a design that is stable will have a sluggish response. The other limitation is the direct coupling of queue length and loss probability, which brings the flows in an overloaded system double penalty of high delay and high loss. They applied the classical control system techniques to design a PI controller to determine the packet drop probability. Figure A4-1 shows the implementation of the PI controller in the RED capable router (called PI-RED in this thesis).

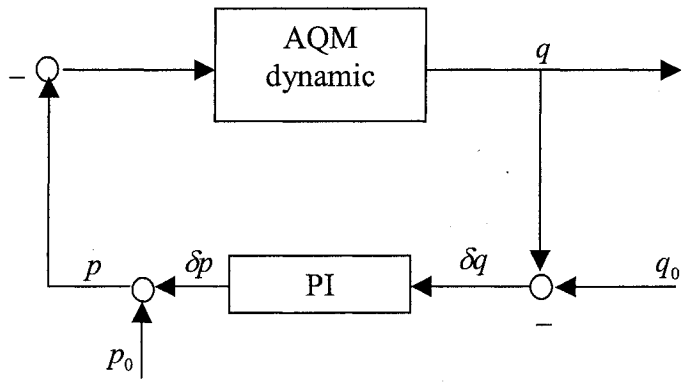


Figure A4-1 Implementation of PI Controller in RED Capable Router

Unlike RED, PI-RED does not determine the packet drop probability directly from the queue length level, but uses the PI controller output to determine the dropping probability as shown in Figure A4-1, where p is the dropping probability assigned to the incoming packet, p_0 is the reference dropping probability, δp is the PI controller output, q is the router buffer

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 the PI controller.

The implementation of PI-RED requires a modification to the RED averaging algorithm and adds two additional state variables. One is the instantaneous queue length at the previous sampling time, the other, the packet drop probability at the previous sampling time. Figure A4-2 shows the PI-RED algorithm.

```
for each packet arrival
      if q ≥ physical memory limit
            drop the packet
      else
            drop the packet with probability p

for each probability sampling time
      p := a·(q − q0) − b·(q_old − q0) + 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:
      q0 :    reference queue size
      a, b : PI controller constants
```

Figure A4-2 PI-RED Algorithm