uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Xuhua OUyang

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE / DEGREE

Department of Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Suboptimal Congestion Control Laws for TCP-RED Dynamics in Computer Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Professor N. Ahmed

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

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

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor Tet Yeap

Professor Oliver Yang

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Suboptimal Congestion Control Laws for TCP-RED Dynamics in Computer Networks

Xuhua Ouyang

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the degree of Master of Science in Systems Science

Faculty of Engineering

School of Information Technology and Engineering

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-49257-4
Our file Notre référence
ISBN: 978-0-494-49257-4

Contents

Acknowledgements

I would like to express my heartfelt gratitude to professor N.U.Ahmed, my thesis supervisor, for his guidance and patience throughout the completion of the thesis and related research work at the University of Ottawa, as well as for his inspiration and understanding at the weekly discussion.

I am also thankful to Mr. Li Cheng for his helpful suggestions at the early period of the project.

Abstract

Based on a dynamic model that simulates the behavior of TCP flows and active queue management (AQM) scheme two specific congestion avoidance modes, feedback control law and real time control law, have been presented respectively in the thesis. The control laws proposed monitor the status of buffers and multiplexor of the router and detect incipient congestion by sending warning signals to the sources.

In our work, the AQM scheme can be modeled as a feedback control system governed by stochastic differential equations driven by a doubly stochastic point process with intensities being the controls. Besides, the AQM scheme can also be modeled as a real time control system in which the state of the router is monitored online and an optimal control must be executed in real time for each of sampled time intervals.

Key Words: Feedback Control, Real time Control, RED, Dynamic Model, TCP, AQM.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background - Congestion Control and Avoidance

Over the recent years, Transmission Control Protocol (TCP) has been one of the most important protocols for data exchange via the Internet [27]. The TCP flow congestion in computer network is a state in which the aggregate demands for network resources arising from TCP connections may exceed the capacity of the systems. Obviously, it is expected that TCP congestion may result in degradation of service quality for the end users. Hence, congestion control and congestion avoidance techniques have been developed to avoid congestion collapse in computer networks [22]. Representative improvements to congestion control and congestion avoidance include TCP Reno and early congestion detection in routers.

Of all several TCP standards, TCP Reno is the most popular implementation. Reno congestion control consists of two major phases: slow start and congestion avoidance. In addition, congestion control is integrated with related fast retransmit and fast recovery error recovery mechanisms. The slow start algorithm is involved in resolving the congestion problem and congestion avoidance is the algorithm that tries to solve the problem with

lost packets. They work together to implement the congestion control mechanism for TCP end-to-end.

Today, "tail drop" queue management [3] has been employed as the implementation of router-based congestion avoidance mechanism. The traditional technique sets a maximum queue size and accepts packets until the maximum size is reached, then drops subsequent incoming packets until the queue decreases. This method has two important drawbacks [3]: (a) lock-out, tail drop allows a single or a few connections to monopolize queue size while preventing other connections from getting space in the queue; (b) full queue, the tail drop discipline allows queues to keep a full status for a long period of time since tail drop sends congestion signals only when the queue has become full.

To remedy the performance degradation of the TCP congestion control over tail drop at routers, "active queue management" (AQM) [3] algorithm has been introduced. AQM drops packets before a queue becomes full so that the sources can make the responses to congestion before buffers overflow. In [3], IETF recommends that Internet routers should implement random early detection (RED) as an AQM mechanism to manage queue sizes.

In a word, a RED router attempts to control the congestion level, limit queuing delays, and avoid buffer overflows by using early packet dropping [4]. Generally, RED gateways require network providers to specify five parameters [4]: the buffer size (Q), the minimum (min_{th}) and maximum (max_{th}) thresholds of the RED region, the maximum dropping probability (max_p), and the weight factor w_q used to calculate the average queue size (q_{avg}). Instead of the instantaneous queue size, RED was specifically designed to use the average queue size as a measure of incipient congestion. Thus the short-term large increases in the queue sizes, resulting from bursty traffic or from transient congestion, do not result in a significant variations in the average queue size. If q_{avg} does not exceed min_{th} a RED router will not drop any packets. Early packet dropping starts when q_{avg} exceeds min_{th}. RED

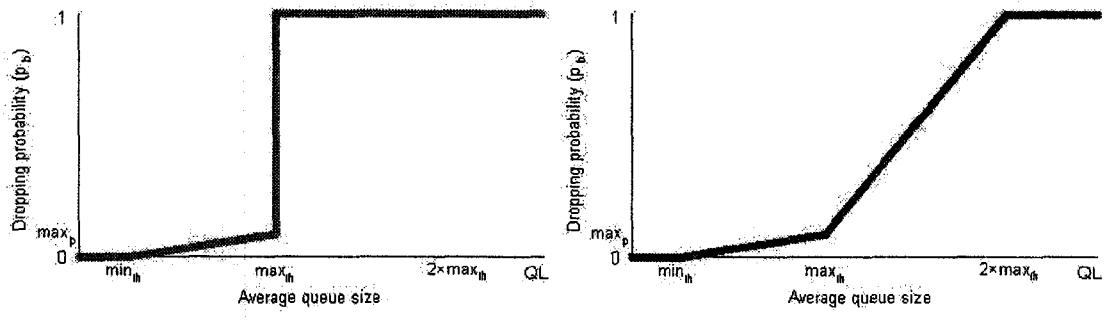
algorithm drops the packet by random when q_{avg} is between min_{th} and max_{th}. This is accomplished by making Bernoulli trials using a probability p_a, which is calculated according to the following formulae [4]:

$$p_b = \frac{max_p \times (q_{avg} - min_{th})}{max_{th} - min_{th}} \tag{1.1}$$

and

$$p_a = \frac{p_b}{1 - count \times p_b} \tag{1.2}$$

where max_p usually is set to 2 % or 10% [4]. *count* is the number of packets since the last packet drop. p_b varies linearly between 0 and max_p, while p_a, the actual packet dropping probability, increases with *count*. When q_{avg} exceeds max_{th}, all packets have to be dropped, shown in Figure 1.1 (a).



(a) Original RED (b) Gentle RED

Figure 1.1: The packet dropping probability (p_b) in RED ($max_p = 10\%$)

Unfortunately, studies [5, 6] show that the original RED algorithm can induce network instability and major traffic disruption if the parameters mentioned above not properly configured. The main reason is that RED configurations that work well for one traffic load may not work well for another. Many recommendations [5, 6, 7, 8, 9, 24] have been proposed

to attempt to fix the problems since RED was presented by Sally Floyd and Van Jacobson as a congestion avoidance mechanism in packet-switched networks in 1993. All RED variants follow a basic principle: AQM mechanism detects incipient congestion by computing the average queue size and notifies connections of congestion either by dropping arriving packets or by setting a bit in packet headers.

Gentle RED (GRED) was proposed in [7] as an improvement to RED, under which packets are dropped with a linearly increasing probability until q_{avg} exceeds $2 \times max_{th}$, after that all packets are dropped, shown in Figure 1.1 (b). In RED, when the average queue size becomes large, the packet drop probability is changed drastically. Hence, RED has a problem that the queue size becomes unstable when the average queue size is large. GRED solves this problem by gently changing the packet drop probability when the average queue size is large. Although extensive studies on RED have been performed by many researchers, the performance of GRED has not been fully investigated [25].

Adaptive RED (ARED) [9] tries to adapt the value of max_p so that the average queue size is halfway between min_{th} and max_{th}, The maximum drop probability, max_p, is kept between 1% and 50% and is adapted gradually. ARED makes RED well tuned over a wider range of traffic conditions, but the determination of the values of min_{th} and max_{th} is an open question.

In [8], authors proposed a modified version of RED governed by doubly stochastic Poisson-driven stochastic differential equations. The dynamic model developed contains a suboptimal feedback control law which observes the status of queue of the (multiplexer) router and adjusts or controls the dropping intensity by sending congestion warning signals to the sources. The strong advantage of the algorithm is that the critical parameters suggested by the original RED AQM are automatically determined by the optimization process based on the current network conditions instead of by the constant configurations. The experimental

results demonstrate that the proposed model (and algorithm) can improve the system performance significantly. However, it was observed that the feedback control law introduces global synchronization into the system since all of the connections share the same mean rate (intensity) of congestion warning signals.

1.2 Thesis Contributions

In this work, we proposed an improved model mainly aiming at solving the problem of global synchronization that results from many connections reducing their windows at the same time while preserving the advantages in [8]. In this modified system we suggest the following configuration: access of each TCP connection to the router is controlled by a distinct buffer (for the connection). Two specific congestion control mechanisms, feedback and real time control, are introduced respectively to act as the congestion control mechanisms.

The centralized feedback control law used in previous models is replaced by a combination of individual feedback control laws for each of the connections. For analysis of the Active Queue Management system, we still use doubly stochastic Poisson-driven stochastic differential equations to model the behaviors of TCP flows and queues of router in which the drop rate (intensity) for each of the connections is an appropriate function of both the average queue size at the multiplexer and average queue size at the buffer dedicated to the connection. The presented objective function which incorporates packet losses and global synchronization in addition to link utilization and congestion is minimized by an iterative choice of every single rate mentioned above. With experimental results, we show the revised AQM approach is not only capable of effectively controlling congestion and synchronization but also more robust than the earlier one.

In addition to the feedback control, an open-loop control called real time control [26]

is proposed as an alternative policy to apply in the context of congestion control of packet switched communication networks under which the feedback control is difficult to apply . The real time control law compute the given cost functional with the approximation of the future state status and execute optimal controls online to protect TCP flows in the Internet from congestion. The technique presented here depends on the availability of sampled state information. At the end of the thesis, comparison of two control laws is given in terms of the system performances based on the experimental results.

1.3 Thesis Organization

The rest of the thesis is organized as follows: In chapter 2, system model is presented. Control laws are given in chapter 3. Objective function and state-space formulation are presented in chapter 4. Numerical results are presented in chapter 5. The paper ends with conclusion in chapter 6.

Chapter 2

System Model

In this chapter, we first introduce the definitions of indicator function and poisson process, and then review dynamical system, the optimal control and characteristics of computer network services. Based on these principles, the system dynamics being developed is modeled.

2.1 Definition

2.1.1 Indicator Function

Let S denote any logical or mathematical statement and define the indicator function as follows:

$$I(S) \equiv \begin{cases} 1 & \text{if } S \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

2.1.2 Poisson Process

Suppose that for any intensity λ_i the components of the poisson process $\{N_i^{\lambda_i}(t), t \in I\}$ are mutually independent, and for $\Delta t > 0$ probability that $N_i^{\lambda_i}$ makes k jumps during the time

interval $[t, t+\Delta t]$ is given by

$$P\{N_i^{\lambda_i}(t+\Delta t) - N_i^{\lambda_i}(t) = k\} = \frac{e^{-\lambda_i \Delta t}(\lambda_i \Delta t)^k}{k!}, \quad k = 0, 1, \ldots \tag{2.1}$$

Hence for almost all $t \in I$, we have

$$P\{N_i^{\lambda_i}(t+\Delta t) - N_i^{\lambda_i}(t) = 1\} = e^{-\lambda_i \Delta t}(\lambda_i \Delta t) \tag{2.2}$$

$$P\{N_i^{\lambda_i}(t+\Delta t) - N_i^{\lambda_i}(t) = 0\} = e^{-\lambda_i \Delta t} \tag{2.3}$$

Combining (2.2) and (2.3), we have

$$P\{N_i^{\lambda_i}(t+\Delta t) - N_i^{\lambda_i}(t) \leq 1\} = (1+\lambda_i \Delta t)e^{-\lambda_i \Delta t} \tag{2.4}$$

For sufficiently small Δt, (2.4) can be considered as

$$P\{N_i^{\lambda_i}(t+\Delta t) - N_i^{\lambda_i}(t) \leq 1\} = 1 \tag{2.5}$$

2.2 Dynamical Systems and The Optimal Control

A deterministic system operating over the fixed interval $t \in I \equiv [0, T]$ is governed by a differential equation of the form

$$\dot{x} = f(x(t), u(t), t), \qquad x(0) = x_0, \tag{2.6}$$

where $x(t)$, the state variable, is determined by $u(t)$, the control variable, and initial state $x(0)$.

The general mode for a stochastic system is given by

$$dx = f(x(t), u(t))dt + g(x(t), u(t))dW(t), \qquad x(0) = x_0 \tag{2.7}$$

where $f : I \times R^n \rightarrow R^n$, $g : I \times R^n \rightarrow R(n \times d)$, the space of $n \times d$ matrices, $\{W(t), t \in I\}$ is a d-dimensional Wiener process [19].

The stochastic differential equation driven by Poisson point process $\{N_i^{\lambda_i}(t), t \in I\}$ with the rate λ_i is given by the model

$$dx = f(x(t), t)dt + g(x(t), t)dN_i^{\lambda_i}(t), \qquad x(0) = x_0 \tag{2.8}$$

where

$$dN_i^{\lambda_i}(t) = \begin{cases} 0, & \text{no event at the time t} \\ 1, & \text{event at time t} \end{cases} \tag{2.9}$$

and λ_i is determined by the choice of $u(t)$. In section 2.4, a stochastic differential model driven by point process like (2.8) is developed for an AQM system with i TCP flows by taking into account different rates λ_i.

The optimal control is the one that minimizes a certain cost functional. The optimal control deals with the problem of finding a control law for a given system so that a certain criterion is achieved .

Given the constraints

$$x(t) \in \Omega_1, \quad u(t) \in \Omega_2, \quad f(x, u, t) \in \Omega_3, \tag{2.10}$$

Ω_1 : state constraints set,

Ω_2 : decision constraints set,

Ω_3 : joint state and decision constraints set, for the objective functional

$$J(u) = \phi(x(T)) + \int_0^T \ell(x(t), u(t), t)dt, \tag{2.11}$$

the optimal control problem is to select $u^*(t)$ which minimize (2.11) subject to (2.8) and (2.10).

2.3 Computer Network Service Characteristics

2.3.1 Round Trip Time

The round trip time (RRT) of a TCP packet is defined as the time it takes for the packet to reach the receiver and to carry the generated acknowledgment to return to the sender [10]. The round trip time packets experience through a network normally is decomposed into four components [2]:

$$RRT = T_{TRANS} + T_{PROP} + T_{QUEUE} + T_{PROC} \tag{2.12}$$

In (2.12) T_{TRANS} is the time required to transmit a packet, so

$$T_{TRANS} = \frac{Packet\ Size}{Transmission\ Speed} \tag{2.13}$$

T_{PROP} is the packet propagation delay, so

$$T_{PROP} = \frac{Distance\ from\ Source\ to\ Destination}{Speed\ of Electrical\ or\ Optical\ Signal} \tag{2.14}$$

T_{QUEUE} is the queuing delay in the router. It occurs whenever the packet rate of traffic coming into the router exceeds the capacity of the outgoing link. The excess packets are queued in the router buffers, so

$$T_{QUEUE} = \frac{Packets\ in\ The\ Queue}{Capacity\ of\ The\ Outgoing\ Link} \tag{2.15}$$

In contrast with T_{TRANS} and T_{PROP}, queuing delay is significantly affected by the network control policy. Finally, T_{PROC} is the processing time required by the network router. In our work, we will assume that T_{TRANS} and T_{PROC} are negligible.

2.3.2 Window Size Adjustment in TCP

The window size is the maximum number of bytes the source can send in a set of packets for which it has not received acknowledgments [2].

The TCP window size adjustment starts by increasing the window size exponentially fast to detect the available transmission rate. When the source fails to get an acknowledgment before timeout, it suspects that a router has dropped a packet and it reduces its window size by 50%. From that point on, TCP uses additive increase multiplicative decrease (AIMD) algorithm [2] to prevent and minimize congestion. In particular, for each round trip time, as long as a source does not detect any packet loss or receive any congestion signal, it keeps on increasing its window size linearly. When it detects a loss or receives a congestion signal, the source halves its window size.

2.4 Modeling of System Dynamics

2.4.1 System Model

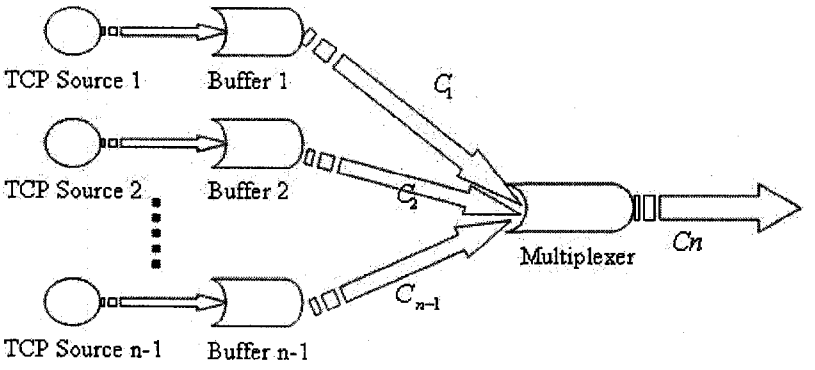


Figure 2.1: System Model

We consider the system presented in Figure (2.1). There are $n-1$ TCP flows, labeled $i =$

$1, 2, \ldots, n-1$, which attempt to access the router. Each individual source, i is connected to a dedicated buffer which communicates with the router through a dedicated link of bandwidth C_i. Actually, in our formulation, the single connection could represent the aggregation of all TCP connections bottlenecked at the same link and having the same round trip time. The packets reaching the router from all the individual sources are multiplexed for onward transmission through an outgoing link of capacity C_n.

2.4.2 Traffic Model

In the absence of drop priorities, the relationship between C_i and C_n may be given by

$$C_i(t) = \beta \cdot \frac{C_n(t)}{n-1}, \quad i = 1, 2 \cdots n-1, \tag{2.16}$$

In reality, $C_n(t)$ is a time variable depending on the current network situations. β is a fixed value set by network providers. The determination of the value of β will be discussed in the chapter 5. Thereby $C_i(t)$ also is a time variable which changes following the change of $C_n(t)$.

For the specific case of Differentiated Services model (DiffServ) where different connections have different priorities, network designers just provide wider bandwidths between buffers and the multiplexer for the connections with lower drop priority and narrower bandwidths for those with higher drop priority.

To develop a simulation of a dynamic system, we must first develop mathematical models of major system components as well as of significant interactions between the system and its input.

2.4.3 Dynamic Models for TCP-RED System

For simplicity, we ignore the slow start and retransmit timer mechanisms. Thus we have a model and analysis during the congestion avoidance mode only.

The dynamic model of the TCP flow control system can be described in terms of window size of the sources and queue sizes at buffers and the multiplexer. The window size is governed by the following equation:

$$dw_i(t) \equiv \frac{1}{R_i(q_i(t), q_n(t))}dt - I(w_i(t) > 0)\frac{w_i(t)}{2}dN_i^{\lambda_i}(t), \quad i = 1, 2 \cdots n-1, \tag{2.17}$$

where $w_i(t)$ denotes the window size of TCP connection i at time $t \geq 0$ and the process $N_i^{\lambda_i}(t)$ represents the number of packets dropped from the connection i over the time interval $[0, t]$. This is a point process with intensity λ_i. The process $q_i(t)$ is the queue size at buffer i and $q_n(t)$ is the queue size at the multiplexer measured at time $t \geq 0$. The expression $R_i(q_i(t), q_n(t))$ is dependent on both $q_n(t)$ and $q_i(t)$, is the round trip time of TCP connection i and the destination. This is generally given by the following expression:

$$R_i(q_i(t), q_n(t)) \equiv a_i + \frac{q_i(t)}{C_i} + \frac{q_n(t)}{C_n}, \quad i = 1, 2 \cdots n-1, \tag{2.18}$$

where a_i denotes the propagation delay between the source and destination and it is considered here as a random variable uniformly distributed over the range $D \subset (0, \infty)$. The set D depends on the network size and its topology and this can be estimated easily while it is impossible to incorporate propagation delay packet by packet and present a reasonably correct analysis of the traffic process. For our numerical experiments we have chosen $\{a_i\}$ to be uniformly distributed independent random variables with values in $D = [0.01, 0.2]$. The lower and the upper limits of the set D is determined from the topology of the network. The remaining part of the expression models the queuing delay for the source i. In summary, the first term on the right hand side of the expression (2.17) gives the windows opening rate and the second the closing rate.

The dynamics of buffers and multiplexer queue loads can be described as follows:

$$dq_i(t) = u_i(t)I(q_i(t) < Q_i)dt - v_i(t)dt, \quad i = 1, 2 \cdots n, \tag{2.19}$$

where Q_i is the size of buffer i and Q_n is the size of the multiplexer (main buffer). The processes $u_i(t)$ and $v_i(t)$ expressed as follows represent the instantaneous rates of incoming and outgoing packets at time t :

$$u_i(t) = \begin{cases} \frac{w_i(t)}{R_i(q_i(t),q_n(t))}, & \text{for } i = 1, 2 \cdots n-1 \\ \\ \sum_{j=1}^{n-1} v_j(t), & \text{for } i = n \end{cases} \tag{2.20}$$

$$v_i(t) = C_i(t)I(q_i(t) > 0) + min\{C_i(t), u_i(t)\}I(q_i(t) = 0), \quad i = 1, 2 \cdots n. \tag{2.21}$$

2.4.4 Time Delay Models

The models presented above are independent of the two-way propagation delay between the router and the sources. In reality there is always a communication delay. Assuming an average delay of δ units of time, equations (2.17) and (2.20) can be modified as follows:

$$dw_i(t) \equiv \frac{1}{R_i(q_i(t), q_n(t)} dt - I(w_i(t) > 0)\frac{w_i(t)}{2} dN_i^{\lambda_i}(t-\delta), \quad i = 1, 2 \cdots n-1 \tag{2.22}$$

$$u_i(t) = \frac{w_i(t-\delta)}{R_i(q_i(t), q_n(t))}, \quad i = 1, 2 \cdots n-1. \tag{2.23}$$

The first equation takes care of delay from router to sources and the second from sources to router.

Chapter 3

Control Strategies

The chapter is organized by three sections. The first section deals with general principles of control mechanisms in control theory and last two sections present an open-loop and a feedback control modes that act as congestion controls in AQM scheme. Here the congestion avoidance mechanism is modeled as an AQM controller under a stochastic counting process based on the state control theory.

3.1 Control Mechanisms

In control theory, there are two basic types of control mechanisms, open-loop and feedback [20]. Open-loop control, also called a non-feedback control, computes its input into a system only by using the current state of the system. A characteristic of the open-loop control is that it does not use feedback to determine if its input has achieved the desired goal. This means that the system does not observe the outputs of the processes that it is controlling [21]. Consequently, a open-loop system can not correct any errors it could make. Figure 3.1 is a block diagram illustrating a open-loop control system.

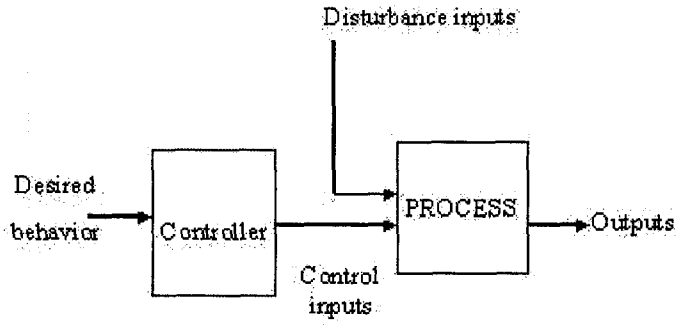


Figure 3.1: Open-loop control system block diagram

To avoid the problems of the open-loop control, control theory introduces feedback. Feedback is a process in which the outputs of a system is fedback to the input [21]. Figure 3.2 shows basic elements of a feedback control system represented by a block diagram. A closed-loop control uses feedback to control outputs of a dynamic system. The system inputs have an effect on the system outputs which is measured by mechanisms and processed by the control. Results are used as the input for the process to close the loop.

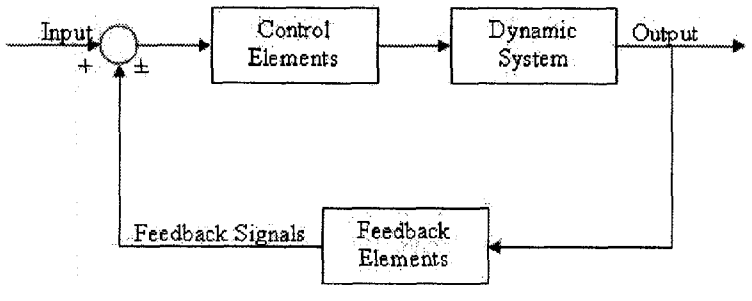


Figure 3.2: Feedback control system block diagram

3.2 Real Time Control Law

We first introduce an open-loop control called the real time control [26] by which the AQM scheme can construct optimal controls in real time to avoid congestion in computer network. The controls must be computed and executed online within every single sampled time interval as the system evolves. The algorithm presented here depends on the availability of sampled state information [26].

We consider the stochastic differential equation (2.8) and the cost functional (2.10). Suppose the interval I is given by the union of k subintervals $\{I_j\}, I \equiv \bigcup_{j=1}^{k} I_j$ where $I_j \equiv [t_{j-1}, t_j), j = 1, 2, \cdots, k$, with $t_0 = 0$ and $t_k = T$. The state is sampled at times $\{t_0, t_2, \ldots, t_{k-1}\}$. We want the controls to be constant on each of these intervals taking values from the compact set U. Given the state $x(t_{j-1})$ at time t_{j-1}, the objective is to choose a control for the interval $[t_{j-1}, t_j)$ that minimizes the cost of running the system over this period.

Considering the first interval I_1 with state information $x(t_0) = x_0$, and any control $\lambda \in U$, the state at time t_1 is given by

$$x(t_1) \approx x_0 + f(t_0, x_0)(t_1 - t_0) + g(t_0, x_0)\lambda, \tag{3.1}$$

and the running cost for the interval I_1 is approximated by the expression

$$J_1(\lambda) \equiv \ell(t_0, \frac{x_0 + x(t_1)}{2}, \lambda)(t_1 - t_0). \tag{3.2}$$

Based on (3.1) and (3.2), we have

$$J_1(\lambda) \equiv \ell(t_0, x_0 + \frac{f(t_0, x_0)(t_1 - t_0) + g(t_0, x_0)\lambda}{2}, \lambda)(t_1 - t_0). \tag{3.3}$$

For sufficiently small intervals (frequent sampling) these approximations may be acceptable in practice. Since f is continuity in the control variable, $\lambda \to x(t_1, \lambda)$ is continuous and hence

continuity of ℓ implies continuity of $J_1(\lambda)$. By compactness of U, J_1 attains its minimum at some $\lambda \in U$. This gives the suboptimal control λ^* for the interval I_1. Using this control one obtains the approximation of the state at time t_1

$$x^*(t_1) \equiv x(t_1, \lambda^*).$$

Thus, in general, given the state $x(t_{j-1})$ at the sampling time t_{j-1}, the cost functional to be minimized for the interval I_j is given by

$$J_j(\lambda^*) \equiv \ell(t_{j-1}, x(t_{j-1}) + \frac{f(t_{j-1}, x(t_{j-1}))(t_j - t_{j-1}) + g(t_{j-1}, x(t_{j-1}))\lambda^*}{2}, \lambda^*)(t_j - t_{j-1}) \quad (3.4)$$

for $j = 1, 2, \cdots, k$. By virtue of compactness of the set U and continuity of $J_j(\lambda_j)$, there exist a λ^* such that

$$J_j(\lambda^*) = \inf_{\lambda \in U} J_j(\lambda) \quad (3.5)$$

This procedure gives the sequence

$$\{x^*(t_{j-1}), \lambda^*, J_j(\lambda^*)\}, \quad j = 1, 2, \cdots, k$$

which may be considered suboptimal.

Clearly, (3.4) and (3.5) provide a very simple recursive algorithm for real time optimization and control given that the process is monitored at each of the time slots indicated. Note that ℓ may contain the sampling and measurement cost. This optimization is expected to improve the performance of the system measured in terms of running cost. More precisely, performance is expected to improve with the increase of sampling frequency. The required frequency of sampling is also dependent on the frequency of variation of the time varying system. Note that the suboptimal control λ^* for the interval I_j and the corresponding optimal cost $J_j(\lambda^*)$ depend only on the state $x(t_{j-1})$.

When applying real time algorithm, we have assumed that all necessary computation can be finished and the suboptimal control λ^* can executed by the system very shortly. It is essential to spend certain amount of time for data processing during each of sampling periods. If only a very small part of the interval I_j, such as $I_j^\rho \equiv [t_{j-1} + \rho))$ with $t_{j-1} + \rho << t_j$ is used for data processing. the algorithm can be expected to be provide improved performance though not optimal.

3.3 Feedback Control Law

Since all system states in feedback controls can be promptly obtained or estimated, a state feedback controller can be easily implemented and can quickly respond to system dynamics. TCP congestion control mechanisms with an AQM scheme at a router can be modeled as a feedback control system as shown in Figure 3.3.

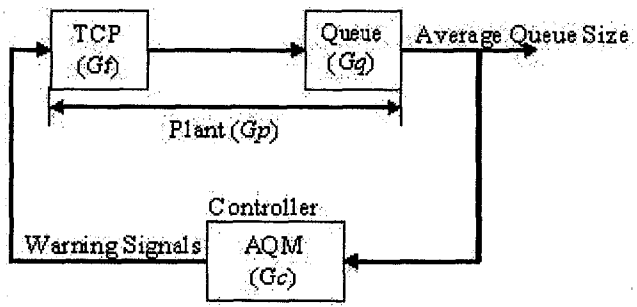


Figure 3.3: TCP-RED as a feedback control system

The implementation of TCP uses warning signals sent by the network as feedback information since warning signals implies congestion occurrence in the network. In this model, the feedback loop consists of a plant (G_p) representing a combination of the end user TCP congestion control mechanisms (G_t) and the router queue dynamics (G_q), and an AQM con-

troller (G_c) which control the generation of warning signals as control signals to the plant. We have examined G_p in the chapter 2 and the emphasis of the this section is on modeling the G_C term for capturing the suboptimal feedback control policy of AQM schemes. A time-delayed feedback control system represented by figure 3.4 will be used in the simulation.

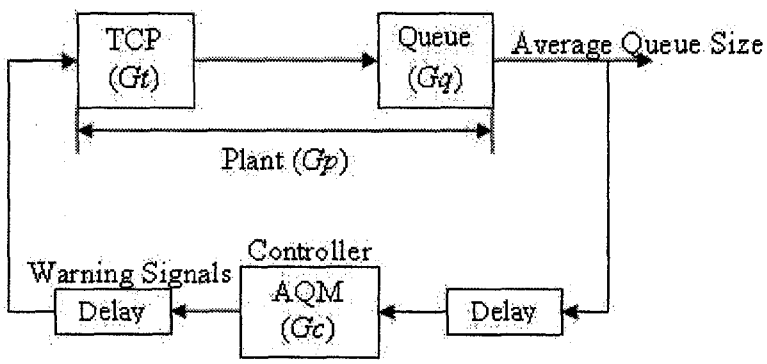


Figure 3.4: TCP-RED as a time-delayed feedback control system

The only information that is readily available to the router is its queue size $q_i, i = 1, 2, \ldots, n$ and possibly its variation with time. One must exercise control on the basis of this information alone. In view of these difficulties, we must simplify the problem and achieve optimal policy. The process $N_i^{\lambda_i}(t)$ in (2.17) is a counting process with intensity process $\lambda_i(t), t \geq 0$. In the context of internet traffic, open loop control is impractical and so out of question. The intensity process is used as a control variable and its choice should depend on the current status of the traffic which means feedback control. Thus, in general, one may choose λ_i as an appropriate nonnegative function of the current state $\xi = \{w_1, w_2, \cdots, w_{n-1}, q_1, q_2, \cdots, q_n\}$ as shown below:

$$\lambda_i(t) \equiv f_i(w_1(t), w_2(t), \cdots, w_{n-1}(t), q_1(t), q_2(t), \cdots, q_n(t)), \quad i = 1, 2 \cdots n-1, \quad t \geq 0, \quad (3.6)$$

In practice it is prohibitively costly to monitor the entire state ξ of the network. The only information that can be easily monitored by the AQM system in the router is the multiplexor queue q_n and possibly the buffer queues $\{q_i\}$. Also it is not practical for the Router to monitor each individual window size. Thus the feedback control for each individual source may be required to depend only on the state of its own buffer and the state of the multiplexor. This leads to a partially observed feedback control law. This is given by the following expression:

$$\lambda_i(t) \equiv f_i(q_i(t), q_n(t)), \quad i = 1, 2 \cdots n-1, \quad t \geq 0. \tag{3.7}$$

In high speed traffic environment the state of queue changes too fast and therefore direct use of this fast changing traffic would result in high frequency fluctuation of the control. To avoid this phenomenon one uses smoothed version of the queue given by the following exponentially weighted moving average [4]:

$$\bar{q}_i(t) = (1 - \kappa_q)\bar{q}_i(t - \Delta t) + \kappa_q q_i(t), \quad t \geq 0, \quad i = 1, 2 \cdots, n \tag{3.8}$$

leading to the following control laws

$$\lambda_i(t) \equiv f_i(\bar{q}_i(t), \bar{q}_n(t)), \quad i = 1, 2 \cdots n-1, \quad t \geq 0. \tag{3.9}$$

Thus the short-term increases in the queue sizes, resulting from bursty traffic or from transient congestion, do not result in significant fluctuation of the average queue size. Since both $\bar{q}_i(t)$ and $\bar{q}_n(t)$ are random process, $N_i^{\lambda_i}(t)$ is a doubly stochastic counting process. The feedback law f_i can be chosen by the system designer as a suitable nonnegative and nondecreasing function defined on $R_+^2 \equiv \{(x, y) \in R^2 : x, y \geq 0\}$. One possible choice of the feedback law f_i is given by:

$$f_i(x, y) = g_i(x, y) I(g_i(x, y) > 0), \quad i = 1, 2 \cdots n-1 \tag{3.10}$$

where, in general, each g_i is given by a bi-variate polynomial of degree m with real valued coefficients such as

$$g_i(x,y) \equiv \sum_{r=0}^{m} \sum_{j=0}^{r} a_{r,j}^i x^j y^{r-j}. \tag{3.11}$$

For numerical experiments we consider a simpler version of this such as

$$g_i(x,y) = a_0^i + a_1^i x + a_2^i y + a_3^i x^2 + a_4^i y^2 + \cdots + a_{2m-1}^i x^m + a_{2m}^i y^m, i = 1, 2 \cdots n-1. \tag{3.12}$$

The variable λ_i actually denotes the mean rate of congestion warnings sent out to the source i. The function f_i, defining the packet dropping scheme for the stream i, is eventually determined by optimization process. This is done by choosing the coefficients of the above polynomials that minimizes the cost functional introduced in the following chapter. Once this is chosen, equation (3.9) determines the intensity and hence the dropping rate. When λ_i equals 0, no packet is dropped. Our feedback model, once optimized, will automatically determine the dropping policy independent of the TCP traffic flow. For numerical simulation using our model, we assume that $N_i^{\lambda_i}(t), t \geq 0$, is a conditional Poisson process, conditioned on λ_i (or doubly stochastic Poisson process).

The major goal of the thesis is to determine an optimal feedback control law out of (3.12) with optimization process.

Chapter 4

Objective Functional and State Space Formulation

This chapter introduces an objective functional based on four performance criteria: TCP flow throughput, system congestion, packet losses and global synchronization. The optimization process is implemented by Monte Carlo method and Random Recursive Search algorithm. State space formulation is presented for the further study.

4.1 Performance Criteria

The main objectives of a system design are to satisfy certain performance criteria. In the context of AQM schemes, the first step is to pose performance criteria. In reality, no one single criteria is enough to indicate whether the performance of a computer network system is acceptable or not. For the AQM we study, performance criteria include TCP throughput, network congestion, packet losses and global synchronization.

1. *TCP throughput:* For efficient use, the queue at the multiplexor should aviod emptiness

to achieve maximum utilization of the link bandwidth C_n.

2. *Network congestion:* For the system under consideration, congestion is considered when the multiplexor is close to its full capacity during certain period of time. we may define the congestion as follows. The system is said to be in the state of congestion when the multiplexor state hits the zone $Q_\alpha \equiv [\alpha Q, Q]$, where $\alpha \in (0, 1)$ and Q is the multiplexor size. In general, the system is considered to be in the state of congestion if 90% of the multiplexor space is occupied. The value of α can be chosen arbitrarily by the network providers. In our case, $\alpha = 0.9$. Choice of $\alpha = 1$ may induce traffic oscillation leading to degradation of system performance.

3. *Packet losses:* The queues at all buffers and the multiplexor should avoid overflow which results in packet losses.

4. *Global synchronization:* The pattern of each sender decreasing and increasing transmission rates at the same time as other senders is referred to as global synchronization [14]. This leads to inefficient use of bandwidth and the large numbers of dropped packets that must be retransmitted.

There are many different ways to measure the performance of a system like each network is different in nature and design. Performance can also be modeled instead of measured. One example of this is using objective functionals to model network performance in a computer network.

4.2 Objective Functional

In the following we use the notation $E\{z\}$ to denote the expected value of any random variable z. The estimation of expected values by the convergent sample average is accomplished

by the Monte Carlo methods [11].

For evaluation of network performance over the running period $I \equiv [0, \tau]$, an objective functional is chosen for feedback control as follows:

$$J(f) \equiv (1/\tau)E\Big\{-J_T + J_C + J_L + J_S\Big\} = -\overline{J}_T + \overline{J}_C + \overline{J}_L + \overline{J}_S \tag{4.1}$$

where

$$f = (f_1, f_2, \ldots, f_{n-1})' \tag{4.2}$$

$$J_T = \int_I \alpha_1(t) C_n(t) I(q_n(t) > 0) dt \tag{4.3}$$

$$J_C = \int_I \alpha_2(t) I(q_n(t) \in Q_\alpha) dt \tag{4.4}$$

$$J_L = \int_I \alpha_3(t) \sum_{j=1}^{n} \Big(u_j(t) I(q_j(t) = Q_j) + max\{0, u_j(t) - q_j(t)\} I(q_j(t) < Q_j) \Big) dt. \tag{4.5}$$

The last component of the expression (4.1) is given by

$$J_S = \int_I \alpha_4(t) I\Big(\sum_{i=1}^{n-1} I(Sign\varphi_i(t) < 0) > \gamma(n-1) \Big) dt \tag{4.6}$$

where $\gamma \in (0, 1)$ is also a design parameter discussed later and

$$\varphi_i(t) = \frac{w_i(t + \Delta t) - w_i(t)}{\Delta t}, i = 1, 2 \cdots n - 1.$$

Here f denotes a feedback control law which has to be chosen to optimize the performance integral given by the expression (4.1). The overall performance of the system is measured in terms of throughput, congestion, packet losses and synchronization. The quantities $\bar{J}_C$, $\bar{J}_L$, $\bar{J}_S$ are respectively the expected costs of congestion, packet losses and synchronization per unit time, while $\bar{J}_T$ is a measure of expected throughput. The quantity J_T gives the utilization of the link bandwidth C_n; J_C is a measure of congestion when the average queue size at

the multiplexer falls into the congestion zone given by $Q_\alpha \equiv [\alpha Q, Q]$ with $\alpha \in (0,1)$. The factor J_L is a measure of packet losses at the multiplexer and the buffers due to overflow suffered during the period $[0,\tau]$. The quantity J_S is a measure of global synchronization. The value of γ in (5.10) can be chosen by network designers in accordance with the number of connections. The real numbers $\alpha_i, i = 1,2,3,4$ are used as relative weights given to each of the costs. These can be chosen by network designers to reflect different concerns and scenarios and assign appropriate weights as necessary. Once all of them are chosen and fixed, one can then attempt to choose the control (or dropping) strategies f (5.7) to minimize the cost functional (4.1). The optimal strategy guarantees maximum expected throughput and minimum expected congestion, packet losses and global synchronization.

4.3 Euler-Maruyama Method

In mathematics, the Euler-Maruyama method is a technique for the approximate numerical solution of a stochastic differential equation [18]. It is named after Leonhard Euler and G. Maruyama.

Considering equation (2.8), we wish to solve this stochastic differential equation on the interval $t \in I \equiv [0,T]$ that can be interpreted in its integral form as follows:

$$x(t) = x_0 + \int_0^t f(x,s)ds + \int_0^t g(x,s)dN^\lambda(s), \quad t \in T. \tag{4.7}$$

The Euler-Maruyama approximation to the solution x follows the principle: the interval $I \equiv [0,T]$ is partitioned into k equal subintervals of width $\tau > 0$:

$$0 = t_0 < t_1 < \ldots < t_k = T \text{ and } \tau = \frac{T}{k},$$

then recursively define $x_i, i = 0,1,\cdots,k-1$ by

$$x_{i+1} = x_i + f(t_i,x_i)\tau + g(t_i,x_i)(N^\lambda(t_{i+1}) - N^\lambda(t_i))$$

for with initial value

$$x_0 = x(0)$$

where we have

$$x_i = x(t_i)$$

for the value of approximation at the discretization time t_i.

4.4 Monte Carlo Method

Monte Carlo methods [11] are a widely used class of computational algorithms for simulating the behavior of various physical and mathematical systems. Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is a method suited to the evaluation of multidimensional average on complex spaces by utilizing many techniques of computer simulation. Then we have:

Theorem : Convergence in probability of the sample average [12]

Let $h(x)$ be a function of random variable with mean $M(h)$ and variance $V(h)$. Let $x_1, x_2, \ldots, x_n$ be a sample of identical independent random variables, and let

$$\tilde{h}_n = \frac{1}{n} \sum_{i=1}^{n} h(x_i)$$

be the sample average, then $\tilde{h}_n$ converges in probability to $M(h)$.

It should be noted that the sample average gets closer and closer to the mean as n increases. This is a critical point, because a numerical estimation will always involve a finite sample size. Since we use Monte Carlo method to calculate the expected values, the number of sample paths required for numerical experiments is crucial and has to be determined carefully. The number is very strongly dependents on the variance of the traffic. The larger the variance, the larger is the number of sample paths required. For numerical computation,

for the given variance, a set of 300 traces of sample paths is good enough for the used traffic [13].

4.5 State Space Formulation

For further analysis, it is convenient to write the state-space model for the system. Denoting the state of the system by $x \equiv (w_1, w_2, \ldots, w_{n-1}, q_1, q_2, \ldots, q_n)'$, one can write the system in the state-space form as follows:

$$dx = F(x)dt + G(x)dN^{\lambda}(t), \tag{4.8}$$

where the vector field $F(x)$ of dimension $2n-1$ is given by

$$F_i(x) = \begin{cases} \frac{1}{R_i(x_{i+n-1}, x_{2n-1})}, & 1 \le i \le n-1 \\ \triangle_i I(x_i < Q_{i-n+1}) - \{C_{i-n+1} I(x_i > 0) + min\{C_{i-n+1}, \triangle_i\} I(x_i = 0)\}, & \\ & n \le i \le 2n-1 \end{cases} \tag{4.9}$$

in which

$$\triangle_i = \begin{cases} \frac{x_{i-n+1}}{R_{i-n+1}(x_i, x_{2n-1})}, & n \le i \le 2(n-1) \\ \Sigma_{k=1}^{n-1}(C_k I(x_{k+n-1} > 0) + min\{C_k, \frac{x_{k+n-1}}{R_k(x_{k+n-1}, x_i)}\} I(x_{k+n-1} = 0)), & i = 2n-1 \end{cases} \tag{4.10}$$

The function $G(x)$ is a $(2n-1) \times (n-1)$ matrix with entries given by

$$G_{i,j}(x) = \begin{cases} I(x_i > 0)(\frac{x_i}{2})\delta_{i,j}, 1 \le i, j \le n-1 \\ 0, n \le i \le 2n-1, 1 \le j \le n-1 \end{cases} \tag{4.11}$$

and

$$N^{\lambda} \equiv (N_1^{\lambda_1}, N_2^{\lambda_2}, \cdots, N_{n-1}^{\lambda_{n-1}})'.$$

Since λ_i is determined by the choice of f_i, we may rewrite the system (4.8) as

$$dx = F(x)dt + G(x)dN^f(t) \tag{4.12}$$

where f is the packet dropping scheme to be determined for optimum performance. Clearly, the cost functional (4.1) can be compactly written as

$$J(f) = (1/T)E\{\int_0^T \ell(t, x(t))dt\}, \tag{4.13}$$

where ℓ denotes the integrand of the expression (4.1). We will use Monte Carlo techniques to compute this cost functional and measures of performance. One of the objectives of the network provider is to improve the system performance by using control strategies (4.2) that minimize the cost functional 4.13.

The principal objective is to determine the optimal feedback law modulated by indicator function as shown in the expression (3.10). The feedback law f of the expression (5.7) based on (8) and (10) is dependent on a vector of coefficients of dimension $(n-1) \times (2m+1)$

$$a_i \in R^{(2m+1)}, i = 1, 2, \cdots, n-1.$$

Rearranging these coefficients in any suitable order one may consider the coefficient vector $a \in R^{(n-1)(2m+1)}$ as the choice variable in the optimization process. So the objective is to choose the coefficient vector a^* that minimizes the cost functionals (4.1) or equivalently (4.13). This is accomplished by the optimization process through the iterative choice of a.

4.6 Random Recursive Search Algorithm

Due to the presence of indicator functions, the cost functional (4.1) is not smooth with respect to the coefficients of the polynomial (3.12). Most traditional optimization methods

require computation of gradients. In a non-smooth situation like we have, those methods do not work very well. An intuitive technique, called Random recursive search (RRS) algorithm [16], appears to be a good choice to solve this kind of problems. [15] shows the computation time (Matlab 6.5 and Pentium 1.8G) for RRS algorithm is five times shorter than that of Simulated Annealing (SA). Thus for high-dimensional problems arising from a large number of users connected to a large network, the RRS algorithm is most suitable.

The idea of the RRS is to use the high efficiency feature of random sampling at initial steps to identify promising areas and then start recursive random sampling processes in these areas which shrink and re-align the sample space to local optima. We describe the details of this algorithm as follows [16].

Given an objective function $f(x)$ on the parameter space D with a range of $[y_{min}, y_{max}]$, the distribution function of objective function values can be defined as:

$$\phi_D(y) = \frac{m(x \in D | f(x) \leq y)}{m(D)} \tag{4.14}$$

where $y \in [y_{min}, y_{max}]$. Basically, (4.14) represents the portion of the points in the parameter space D whose function values are smaller than a certain level y. $\phi_D(y)$ is a monotonously increasing function of y in $[y_{min}, y_{max}]$, its maximum value is 1 when $y = y_{max}$ and its minimum value is $m(x_*)/m(D)$ where x_* is the set of global optima. Without loss of generality, assume that $f(x)$ is a continuous function and $m(x \in D | f(x) = y) = 0, \forall y \in [y_{min}, y_{max}]$, then $\phi(y)$ will be a monotonously increasing continuous function with a range of [0,1]. Assuming a $y_r \in [y_{min}, y_{max}]$ such that $\phi_D(y_r) = r, r \in [0, 1]$, a r-percentile set in the parameter space D can be defined:

$$A_D(r) = \{x \in D | f(x) \leq y_r\}$$

Note that $A_D(1)$ is just the whole parameter space D and $lim_{\epsilon \longrightarrow 0} A_D(\epsilon)$ will converge to the global optima. Suppose the sample sequence generated by n steps of random sampling is

$x_i, i = 1, 2, \cdots, n$ and $x^n_{(1)}$ is the one with the minimum function value, then the probability of $x^n_{(1)}$ in $A_D(r)$ is

$$P(x^n_{(1)} \in A_D(r)) = 1 - (1 - r)^n = p$$

Alternatively, the r value of the r-percentile set that $x^n_{(1)}$ will reach with probability p can be represented as:

$$r = 1 - (1 - p)^{1/n}$$

For any probability $p < 1$, r will tend to 0 with increasing n. This means random sampling will converge to the global optima with increasing number of samples. [16] shows that the r-percentile set that n steps of random sampling can reach with a probability of 99% and that random sampling is highly efficient at initial steps since r decreases exponentially with increasing n, and its inefficiency is from later samples. As shown in [16], it takes only 44 samples to reach a point in $A_D(0.1)$ area, whereas all future samples can only improve r value of $x^n_{(1)}$ at most by 0.1.

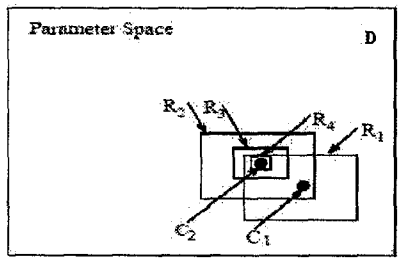


Figure 4.1: Shrink and re-align procedure of RRS

An sample search using RRS is illustrated in Figure 4.1. First, a number of random samples, say n, are taken from the parameter space D. The best point is taken as the center C_1 of the promising region R_1 which is further explored. The point C_1 falls in $A_D(r)$ with probability p. The size of R_1 is taken to be the size of $A_D(r)$ so as to cover at least one

local optimum in $A_D(r)$ with high probability. Then, another l random samples are taken from R_1. Note that l should be much less than n since the search is in a promising area and expects to and better points quickly. If a better point is found within these l samples, the center of the sample space is moved to this point and the size is kept unchanged. For example, in Figure 4.1 the center is moved to C_2, the region R_2 is used as the next sample space. If a better point is not found in l samples, the size of sample space is reduced by half and the center is kept unchanged. In Figure 4.1, R_3 is used as the next sample space after l unsuccessful samples in R_2, and the center C_2 is left unchanged. This shrink-and-re-align procedure is repeated until the size of the region is reduced below a threshold, then the above search process is restarted until stopping criterion is satisfied.

Chapter 5

Numerical Results

Now, we shall demonstrate that the revised version of RED mechanism is more practical and robust than the earlier one in [8], and furthermore seek to validate the effectiveness of feedback control and real time control described above and compare their performance in the same scenario. To this aim, numerical results are produced in this chapter based on the simulation and optimization programs. Experiments in the thesis are carried out on the platform of Pentium 4 2.4G and windows XP SP2 operating environment. The simulation program is Matlab 6.5 R13.

5.1 Numerical Method Used

In the experiments, the TCP-RED congestion control system developed is modeled as a discrete time feedback control system. We use Euler-Maruyama Discrete Approximation (EMDA) Method to simulate the stochastic differential equation. For numerical solution in equation 2.8, the interval $I \equiv [0, T]$, is discretized into n subintervals as $0 = t_0 < t_1 < \ldots < t_k = T$ and $\Delta t = t_1 - t_0 = t_2 - t_1 = \cdots = t_k - t_{k-1}$. Using the EMDA method this equation

is approximated as follows:

$$x(t_j) \approx x(t_{j-1}) + f(t_{j-1}, x(t_{j-1}))\Delta t + g(t_{j-1}, x(t_{j-1}))(N_i^{\lambda_i}(t_j) - N_i^{\lambda_i}(t_{j-1})), \quad j = 1, 2, \ldots, k \tag{5.1}$$

Again, for numerical computation, equation (2.17) can be rewritten as

$$w_i(t_j) - w_i(t_{j-1}) \approx \frac{1}{R_i(q_i(t_{j-1}), q_n(t_{j-1}))}\Delta t - I(w_i(t_{j-1}) > 0)\frac{w_i(t_{j-1})}{2}(N_i^{\lambda_i}(t_j) - N_i^{\lambda_i}(t_{j-1})), \quad j = 1, 2, \ldots, k \tag{5.2}$$

According to equation (2.5), for sufficiently small $\Delta t = t_j - t_{j-1}$, $N_i^{\lambda_i}(t_j) - N_i^{\lambda_i}(t_{j-1})$ in equation (5.2) can be considered as 0 or 1. In other words, at most one warning signal is allowed to be sent to the source to reduce the window size when AQM system detects incipient congestion over the subinterval Δt.

5.2 Simulation Environment Setup

We consider a scenario that the system is comprised of three TCP flow connections and will examine the behavior of these control schemes under a variety of network topologies. The basic system parameters used for numerical simulation are buffer sizes $\{Q_1, Q_2, Q_3, Q_4\}$, weighting factor κ_q, outgoing link capacity C_4, size of the time slot Δt, and three other factors $\{\gamma, \delta, \beta\}$ representing synchronization as defined in (4.6), average time delay appearing in Eq. 2.22, 2.23 and capacity allocation factor as shown in the expression (2.16) respectively. Each simulation lasts 10 seconds (10000 ms).

As a matter of fact, β balances the traffic between the buffers and the multiplexer. Specifically, if β is set too low, sources are assigned less bandwidth thereby limiting their flows and causing packet losses at the user buffers. This may result in reduced utilization of the system significantly. For $\beta \leq 1$, no queue build-up occurs at the multiplexor. In

Δt	κ_q	C_4	Q_1, Q_2, Q_3	Q_4	γ	δ	β
0.001s	0.02	2M packets	40K packets	200K packets	$\frac{1}{3}$	1	2

Table 5.1: System configuration and parameters

contrast, if β is too large more packet losses and congestion occur at the multiplexor. As a compromise we set β equal to 2 in our experiments.

To create realistic congestion phenomenon, all sources keep on increasing their window sizes till they receive warning signals from router and none of them is shut down during the period of simulation.

5.3 Optimization Process

5.3.1 Linear Feedback

For linear state feedback control law we have chosen has the following general form

$$\lambda_i = f_i(\bar{q}_i, \bar{q}_n) = (a_0^i + a_1^i \bar{q}_i + a_2^i \bar{q}_n) I(a_0^i + a_1^i \bar{q}_i + a_2^i \bar{q}_n > 0), \qquad i = 1, 2, 3.$$

The optimization process is started by randomly selecting a coefficient vector $a \in R^9$ which, for convenience, is written in the matrix form,

$$a = \begin{Bmatrix} a_0^1 & a_1^1 & a_2^1 \\ a_0^2 & a_1^2 & a_2^2 \\ a_0^3 & a_1^3 & a_2^3 \end{Bmatrix} = \begin{Bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -3 & -3 & -3 \end{Bmatrix},$$

where the $i-th$ row represents the coefficients of the feedback law of source i. Using the RRS algorithm given in [16], we minimize the cost functional (4.1) by iteratively choosing an appropriate a. If a choice decreases the value of the cost functional, it is accepted and

the corresponding cost is retained. The process is terminated once the stopping criterion is satisfied. Based on this optimization process, the optimum vector obtained for linear feedback control law is given by

$$a^* = \begin{Bmatrix} a_0^{1*} & a_1^{1*} & a_2^{1*} \\ a_0^{2*} & a_1^{2*} & a_2^{2*} \\ a_0^{3*} & a_1^{3*} & a_2^{3*} \end{Bmatrix} = \begin{Bmatrix} -1.83 & 0.027 & 0.012 \\ -2.72 & 0.039 & 0.019 \\ -2.13 & 0.033 & 0.015 \end{Bmatrix}$$

which is able to keep the value of cost functional (4.1) to a minimum. In order to prevent the RRS from getting trapped in a local optima, we have tried with very large perturbation of the parameter a from the starting point and also (the local optimum a^*) with two different starting vectors denoted by $\bar{a}$ and $\tilde{a}$ as given below:

$$\bar{a} = \begin{Bmatrix} a_0^1 + 1000 & a_1^1 + 1000 & a_2^1 + 1000 \\ a_0^2 - 1000 & a_1^2 - 1000 & a_2^2 - 1000 \\ a_0^3 + 1000 & a_1^3 + 1000 & a_2^3 + 1000 \end{Bmatrix} = \begin{Bmatrix} 999 & 999 & 999 \\ -998 & -998 & -998 \\ 997 & 997 & 997 \end{Bmatrix}$$

and

$$\tilde{a} = \begin{Bmatrix} a_0^1 - 1000 & a_1^1 - 1000 & a_2^1 - 1000 \\ a_0^2 + 1000 & a_1^2 + 1000 & a_2^2 + 1000 \\ a_0^3 - 1000 & a_1^3 - 1000 & a_2^3 - 1000 \end{Bmatrix} = \begin{Bmatrix} -1001 & -1001 & -1001 \\ 1002 & 1002 & 1002 \\ -1003 & -1003 & -1003 \end{Bmatrix}.$$

Clearly these are very far from the initial choice of a. Running the RRS algorithm once again with these vectors as the starting points we arrive at the following optimal vectors

$$\bar{a}^* = \begin{Bmatrix} -1.80 & 0.025 & 0.013 \\ -2.69 & 0.036 & 0.022 \\ -2.15 & 0.031 & 0.018 \end{Bmatrix}$$

and

$$\tilde{a}^* = \begin{Bmatrix} -1.82 & 0.023 & 0.010 \\ -2.72 & 0.034 & 0.024 \\ -2.18 & 0.025 & 0.017 \end{Bmatrix}$$

which are very close to a^*. This shows that the RRS algorithm is quite efficient, it is able to escape traps of local minima.

5.3.2 Nonlinear Feedback

The general form of the nonlinear state feedback control law is taken from the following class of polynomials,

$$\lambda_i = f_i(q_i, q_n) = (a_0^i + a_1^i q_i + a_2^i q_n + \cdots + a_{2m-1}^i q_i^m + a_{2m}^i q_n^m) I(a_0^i + a_1^i q_i + a_2^i q_n + \cdots + a_{2m-1}^i q_i^m + a_{2m}^i q_n^m > 0).$$

Our objective is to find the best one. To reduce computational time we have concentrated on second order polynomials, $m = 2$. Again, we use the RRS algorithm to determine the best feedback law from the class of polynomials of degree 2. Randomly we start with two starting vectors $b \in R^{3\times 5} = R^{15}$ as given below:

$$b = \begin{Bmatrix} b_0^1 & b_1^1 & b_2^1 & b_3^1 & b_4^1 \\ b_0^2 & b_1^2 & b_2^2 & b_3^2 & b_4^2 \\ b_0^3 & b_1^3 & b_2^3 & b_3^3 & b_4^3 \end{Bmatrix} = \begin{Bmatrix} -1 & -1 & -1 & -1 & -1 \\ 2 & 2 & 2 & 2 & 2 \\ -3 & -3 & -3 & -3 & -3 \end{Bmatrix}$$

and

$$\bar{b} = \begin{Bmatrix} b_0^1 + 1000 & b_1^1 + 1000 & b_2^1 + 1000 & b_3^1 + 1000 & b_4^1 + 1000 \\ b_0^2 - 1000 & b_1^2 - 1000 & b_2^2 - 1000 & b_3^2 - 1000 & b_4^2 - 1000 \\ b_0^3 + 1000 & b_1^3 + 1000 & b_2^3 + 1000 & b_3^3 + 1000 & b_4^3 + 1000 \end{Bmatrix}$$

$$= \begin{Bmatrix} 999 & 999 & 999 & 999 & 999 \\ -998 & -998 & -998 & -998 & -998 \\ 997 & 997 & 997 & 997 & 997 \end{Bmatrix}$$

The optimum vectors obtained for the quadratic feedback control law are given by

$$b^* = \begin{Bmatrix} b_0^{1*} & b_1^{1*} & b_2^{1*} & b_3^{1*} & b_4^{1*} \\ b_0^{2*} & b_1^{2*} & b_2^{2*} & b_3^{2*} & b_4^{2*} \\ b_0^{3*} & b_1^{3*} & b_2^{3*} & b_3^{3*} & b_4^{3*} \end{Bmatrix} = \begin{Bmatrix} -1.826 & 0.0285 & 0.0110 & 0.00021 & 0.0000023 \\ -2.762 & 0.0400 & 0.0144 & 0.00039 & 0.0000019 \\ -2.154 & 0.0343 & 0.0123 & 0.00032 & 0.0000042 \end{Bmatrix}$$

and

$$\bar{b}^* = \begin{Bmatrix} \bar{b}_0^{1*} & \bar{b}_1^{1*} & \bar{b}_2^{1*} & \bar{b}_3^{1*} & \bar{b}_4^{1*} \\ \bar{b}_0^{2*} & \bar{b}_1^{2*} & \bar{b}_2^{2*} & \bar{b}_3^{2*} & \bar{b}_4^{2*} \\ \bar{b}_0^{3*} & \bar{b}_1^{3*} & \bar{b}_2^{3*} & \bar{b}_3^{3*} & \bar{b}_4^{3*} \end{Bmatrix} = \begin{Bmatrix} -1.254 & 0.0346 & 0.0546 & 0.00001 & 0.0000042 \\ -5.141 & 0.0058 & 0.0173 & 0.00058 & 0.0000052 \\ -3.541 & 0.0658 & 0.0211 & 0.00023 & 0.0000055 \end{Bmatrix}.$$

It is interesting to note that the last two columns of the above matrices are very small. In other words the nonlinearity is negligible. The effectiveness of linear control law is close to that of the quadratic law. However we note a small difference in the two quadratic laws determined by b^* and $\bar{b}^*$.

5.4 Sensitivity Analysis

In general, the control performance of an AQM algorithm is affected by several network parameters such as the buffer size (Q_i), the link capacity (C_n), the synchronization factor (γ), the traffic balance factor (β), the time delay of control (δ), and the round trip time (RTT). Since the buffer size, the link capacity, the synchronization factor, and the traffic balance factor are fixed when an AQM is installed in a router, these parameters are static (i.e., time-invariant) factors. In contrast, the time delay of control and the RTT are dynamic (i.e.,

time-varying) factors because they are changing dynamically over time. Thus, it is important for an AQM algorithm to have adaptive and robust control performance for the dynamic factors. In this section, we examine the sensitivity of the control performance of the optimal control we obtained and RED to the changes of network environments, particularly on the traffic balance factor and the time delay of control, and furthermore discuss how tunable parameters should be set to optimize the system.

5.4.1 Sensitivity to The Traffic Balance Factor

It is important for an AQM algorithm to have adaptive and robust control performance for dynamic TCP flows. We examine the robustness of the optimal feedback control laws obtained above with respect to variations in the network parameters and discuss how tunable parameters should be set to optimize the system.

In this set of experiments, the parameter setup is the same as in the above experiments, except the value of β. For $\beta = 3$, the optimization process is started from

$$a = \begin{Bmatrix} a_0^1 & a_1^1 & a_2^1 \\ a_0^2 & a_1^2 & a_2^2 \\ a_0^3 & a_1^3 & a_2^3 \end{Bmatrix} = \begin{Bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -3 & -3 & -3 \end{Bmatrix},$$

The optimum parameter vector $\hat{a}^*$ was found as

$$\hat{a}^* = \begin{Bmatrix} \hat{a}_0^{1*} & \hat{a}_1^{1*} & \hat{a}_2^{1*} \\ \hat{a}_0^{2*} & \hat{a}_1^{2*} & \hat{a}_2^{2*} \\ \hat{a}_0^{3*} & \hat{a}_1^{3*} & \hat{a}_2^{3*} \end{Bmatrix} = \begin{Bmatrix} -1.78 & 0.023 & 0.011 \\ -2.70 & 0.041 & 0.020 \\ -2.18 & 0.035 & 0.014 \end{Bmatrix}$$

In fig 5.1 we have plotted, for two different values of β, the convergence of the optimization algorithm against the number of iterations.

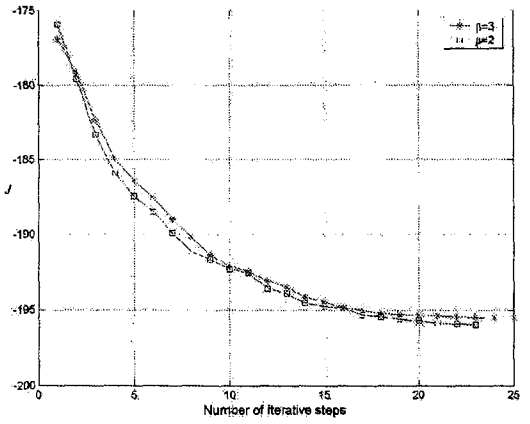


Figure 5.1: the number of iterations for different values of β

β	1.2	1.5	2	3	4	6	8	10
(Linear)	-176.96	-185.21	-195.95	-195.44	-195.22	-195.12	-195.11	-195.15
J(Quadratic)	-175.82	-185.20	-195.94	-195.42	-195.19	-195.18	-195.12	-195.15
(Linear)	181.21	189.09	199.50	199.60	200	200	200	200
$\overline{J}_T$(Quadratic)	180.02	188.99	199.42	199.50	199.92	200	200	200
(Linear)	0.11	0.27	0.38	0.45	0.53	0.57	0.58	0.58
$\overline{J}_C$(Quadratic)	0.10	0.27	0.37	0.43	0.53	0.56	0.57	0.58
(Linear)	4.14	3.61	3.16	3.69	4.21	4.27	4.28	4.24
$\overline{J}_L$(Quadratic)	4.09	3.52	3.10	3.63	4.16	4.23	4.27	4.24
(Linear)	0	0.0035	0.010	0.019	0.026	0.029	0.030	0.030
$\overline{J}_S$(Quadratic)	0	0.0038	0.011	0.020	0.027	0.029	0.030	0.030

Table 5.2: Costs for different values of β

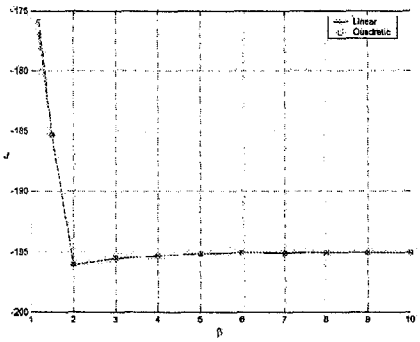
Given $\gamma = \frac{1}{3}, \delta = 1$, we study the variation of performance for different values of β. For the experiments, we have

$$C_i(t) = \beta \cdot \frac{C_4(t)}{3}, \quad i = 1, 2, 3.$$

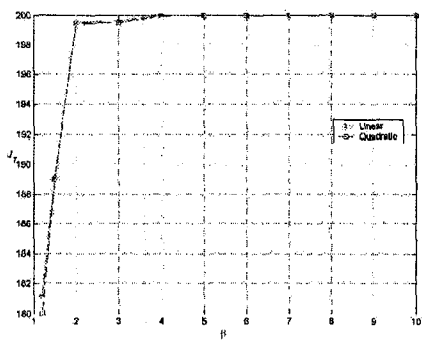
The expected costs for different values of β, such as $\{1.2, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, are computed by use of Monte Carlo technique and shown in table 5.2.

Figure 5.2 shows the comparison of expected costs per unit time of individual factors such as throughput, congestion, packet losses and synchronization for different values of β. As shown in figure 5.2(b), RED (almost) fully utilizes the bandwidth of the outgoing link as β approaches 4. This is because the queue at the multiplexor is seldom empty. The incoming TCP (traffic) flows always keep the queue full. This in turn leads to more congestion and more packet losses at the multiplexor as shown in figure 5.2(c) and (d). Consequently, AQM system would generate warning signals more frequently to force the sources to cut their window sizes more often, which may result in more global synchronization as shown in Fig.5.2(e). Despite the variation of the individual costs the total system cost, as shown in figure 5.2(a), and hence the overall performance is robust with respect to variation of the parameter β beyond 2.

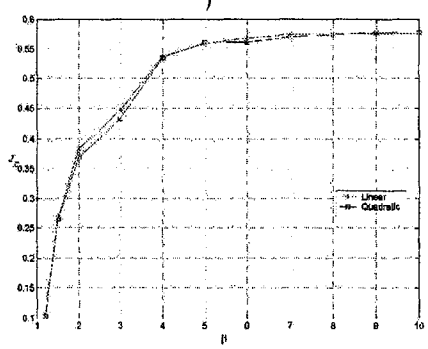
Figure 5.3 shows variations of queues at the buffers and the multiplexor for values of $\beta = 1, 1.2, 2, 10$ corresponding to the linear feedback control law. It was observed that if $\beta > 2$, queue build at the buffers is substantially reduced over the simulation period. Note that Figure 5.2(a) also shows that for identical network conditions, the cost corresponding to the linear feedback policy is almost the same as that of the quadratic policy, both of them reaching the minimum at the point $\beta = 2$. The proposed feedback control algorithm shows the robustness in control performance subject to changes of network environments. This ensures robustness in stability and good performance for a wider range of network parameter uncertainties.



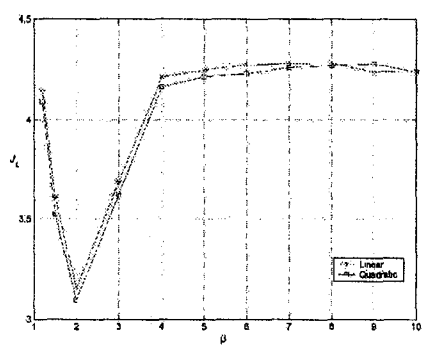
(a) Total cost J



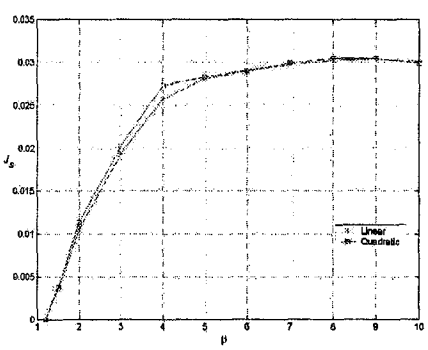
(b) Throughput $\overline{J}_T$



(c) Congestion $\overline{J}_C$

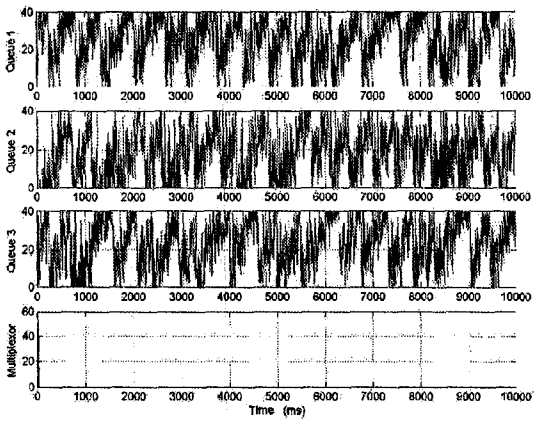


(d) Packet Losses $\overline{J}_L$

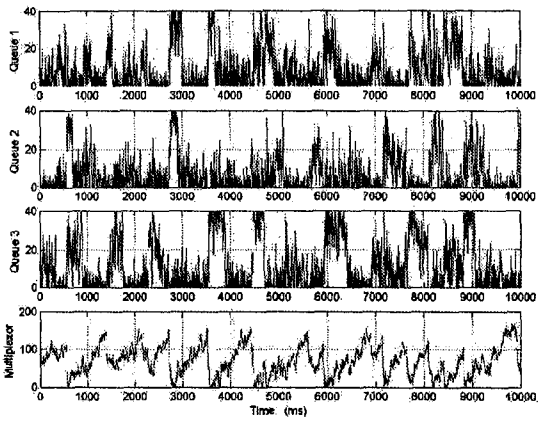


(e) Synchronization $\overline{J}_S$

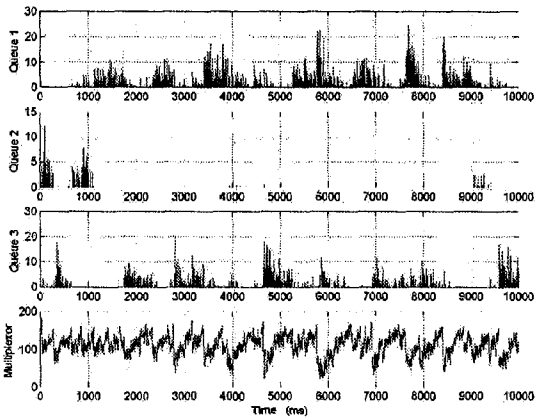
Figure 5.2: Individual costs as functions of β



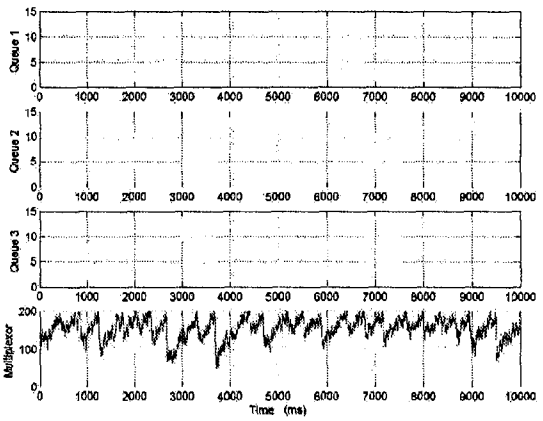
(a) $\beta = 1$



(b) $\beta = 1.2$



(c) $\beta = 2$



(d) $\beta = 10$

Figure 5.3: Variations of queues for different values of β (Linear Case)

5.4.2 Sensitivity to The Time Delay of Control

In fact, the propagation delay is a function of several factors such as the physical distance between sources, destinations and the router including traffic conditions. If one wants to be exact, the problem becomes mathematically very difficult and intractable. Therefore, for simplicity one may view δ as the mean of a random variable having uniform distribution (considering the worst case/maximum entropy) over a compact interval $\Delta \subset [0, \infty)$. For our experiments we have only considered a fixed set of time delays measured in units of basic time slots (used for computing the solutions of the system equations). Given $\gamma = \frac{1}{3}, \beta = 2$, we use $\delta = \{1\Delta t, 3\Delta t, 5\Delta t\}$. The results shown in table 5.3 and figure 5.4 correspond to the optimum linear and quadratic control laws with a^* and b^* as given in sections (5.1.1)-(5.1.2). It is clear from these results that time delay in controls has significant negative impact on system performance. An increase of time delay not only degrades the control performance of an AQM system but may also lead to instability. Thus, the effect of propagation delay of warning signals should be taken into account in designing a robust AQM system.

	J	$\overline{J}_T$	$\overline{J}_C$	$\overline{J}_L$	$\overline{J}_S$
Linear with 1 interval delay	-195.95	199.50	0.38	3.16	0.010
Quadratic with 1 interval delay	-195.94	199.42	0.37	3.10	0.011
Linear with 3 interval delay	-190.58	197.91	0.29	7.03	0.011
Quadratic with 3 interval delay	-190.10	197.38	0.29	6.98	0.012
Linear with 5 interval delay	-186.81	195.16	0.40	7.93	0.012
Quadratic with 5 interval delay	-186.74	194.98	0.40	7.83	0.013

Table 5.3: Variation of Costs with time delay

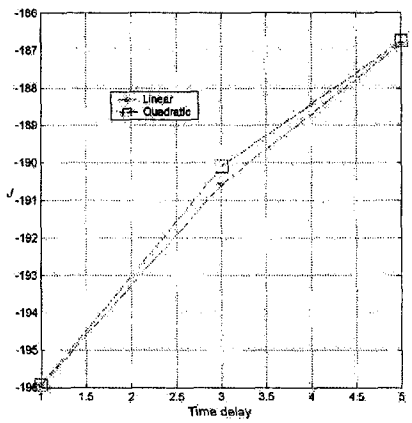
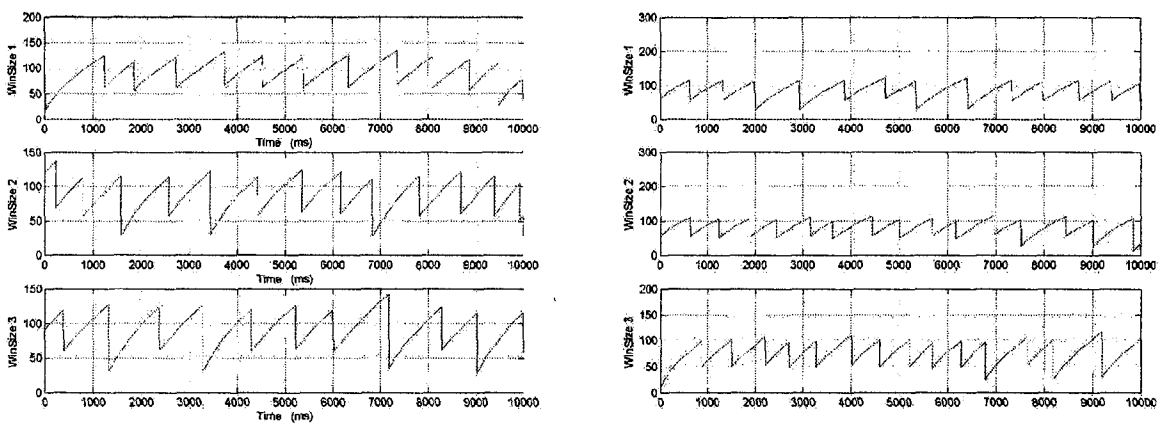


Figure 5.4: Total cost J as a function of time delay



(a) Linear Control Law (b) Quadratic Control Law

Figure 5.5: Variations of windows sizes

5.5 Summary of Feedback Control

Figure 5.5 plots the variations of windows sizes of source 1, 2 and 3 over the experiment period showing global synchronization is reduced effectively as a^* and b^* are applied to the feedback laws f_1, f_2, f_3. The reason is that values of $\lambda_1, \lambda_2, \lambda_3$, which are the functions of average queue sizes at the multiplexer and buffer 1, 2, and 3 respectively, differ with one another due to the difference of coefficients elements in the optimum vector (a^* or b^*), shown in figure 5.6. This ensures the different connection has different dropping rate over simulation period and thereby avoids global synchronization with several connections reducing their windows at the same time.

We notice that in the vector b^*, the elements values of $b_0^{i*}, b_1^{i*}, b_2^{i*}, i = 1, 2, 3$ approximate to $a_0^{i*}, a_1^{i*}, a_2^{i*}$ in the vector a^* respectively and others, i.e. $b_3^{i*}, b_4^{i*}, i = 1, 2, 3$, are very close to zero. This is because b_3^{i*}, b_4^{i*} are coefficients of the second degree of average queue sizes which are definitely very large numbers. In fact, the simulation results achieved by non-linear feedback control laws are almost same as those achieved by linear feedback control law. But non-linear feedback control laws take much longer computation time than linear case however does not compute the system performance drastically. In contrast, linear feedback control law is relatively simpler and more efficient. Therefore, we conclude that non-linear feedback control laws does not exist in (3.9) and the linear case is considered as the optimal feedback control law. Consequently, (3.12) can be rewritten as

$$g_i(x, y) = a_0^i + a_1^i x + a_2^i y, \qquad i = 1, 2 \cdots n - 1, \tag{5.3}$$

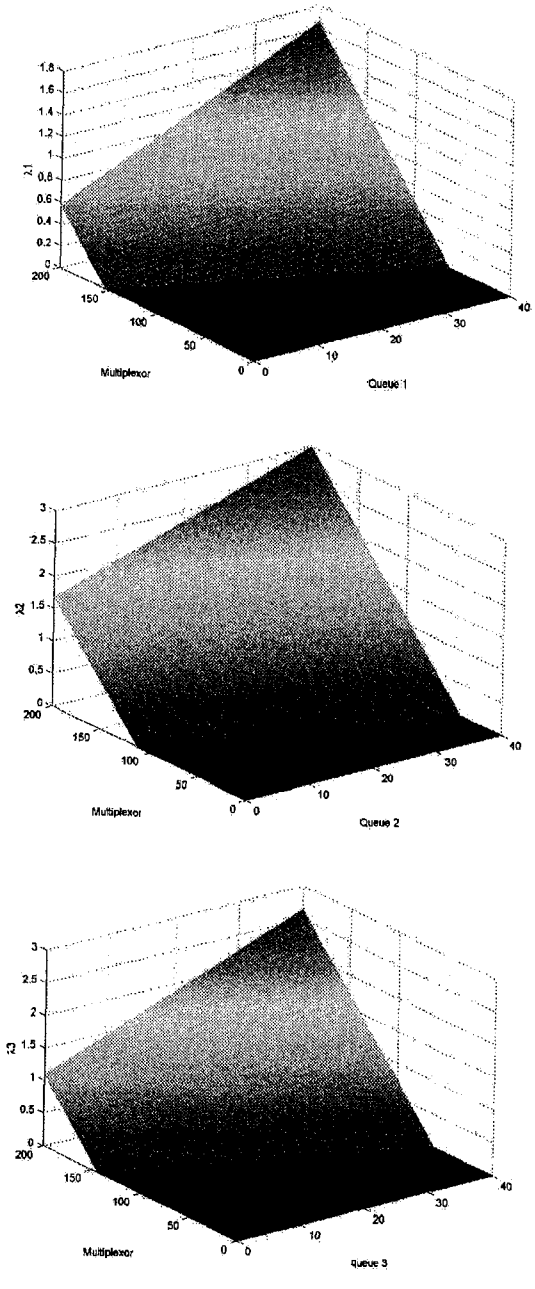


Figure 5.6: λ_i is the function of q_i and q_n

5.6 Experimental Results of Real Time Control

We recall that the real time control system must take an online control for every single time interval based on the computation of the cost functional, while the feedback control system gives the optimal configuration parameters based on the results of optimization process. In next numerical experiments, we compute the corresponding performance of the real time controller and then compare the results with those corresponding to the feedback controller.

5.6.1 Objective Functional

Consider the real time optimization of TCP congestion control governed by the stochastic differential equation (4.8) with the cost functional given by (4.13). Suppose the system state x is monitored at each of intervals $I_j \equiv [t_{j-1}, t_j), j = 1, \ldots, k$. Given the state $x(t_{j-1})$ at time t_{j-1}, the approximation of the true state at time t_j is given by the random vector

$$x(t_j) \approx x(t_{j-1}) + f(t_{j-1}, x(t_{j-1}))(t_j - t_{j-1}) + g(t_{j-1}, x(t_{j-1}))\lambda, \quad j = 1, \ldots, k \tag{5.4}$$

Here the control $\lambda = N(t_j) - N(t_{j-1})$, is a command asking each of sources $1, 2, \ldots, n-1$ for adjustments of windows Size: "cut or not cut". In our case, $\lambda \in U \equiv \{1, 0\}^{n-1}$ and the optimal control for the time interval $I_j \equiv [t_{j-1}, t_j)$ is given by λ_j^* that minimizes the cost functional (5.6). Hence

$$J_j(\lambda_j^*) = \inf_{\lambda \in U} \{\ell(t_{j-1}, x(t_{j-1}) + \frac{1}{2}(f(t_{j-1}, x(t_{j-1}))(t_j - t_{j-1}) + g(t_{j-1}, x(t_{j-1}))\lambda))\}, \tag{5.5}$$

for $j = 1, \ldots, k$. The principal objective is to determine the optimal control λ_j^* for every single time interval $I_j \equiv [t_{j-1}, t_j), j = 1, \ldots, k$, over simulation period I. Here the state of the system can be monitored by the AQM scheme is the multiplexor queue q_n and possibly the buffer queues q_i.

For comparison we also simulate the real time AQM scheme with the same dynamic models (in section 2.4) and network configuration environments (in section 6.2) used by the feedback control system. For evaluation of network performance over the running period $I \equiv \bigcup_{j=1}^{k} I_j$ and for any time interval I_j, the objective functional

$$J_j(I_j, \lambda) \equiv -J_T + J_C + J_L + J_S \tag{5.6}$$

in place of (4.1). where

$$J_T = \int_{I_j} \alpha_1(t) C(t) I(q_n(t) > 0) dt \tag{5.7}$$

$$J_C = \int_{I_j} \alpha_2(t) I(q_n(t) \in Q_\alpha) dt \tag{5.8}$$

$$J_L = \int_{I_j} \alpha_3(t) \sum_{i=1}^{n} \Big(u_i(t) I(q_i(t) = Q_i) + max\{0, u_i(t) - q_i(t)\} I(q_i(t) < Q_i) \Big) dt \tag{5.9}$$

and

$$J_S = \int_{I_j} \alpha_4(t) I\Big(\sum_{i=1}^{n-1} I(Sign\varphi_i(t) < 0) > \gamma(n-1) \Big) dt \tag{5.10}$$

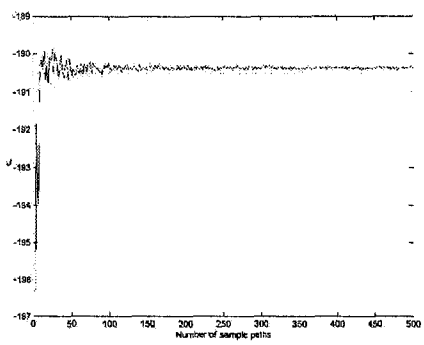
where $\gamma \in (0, 1)$ and

$$\varphi_i(t)) = \frac{w_i(t + \Delta t) - w_i(t)}{\Delta t}, i = 1, 2 \cdots n - 1,$$

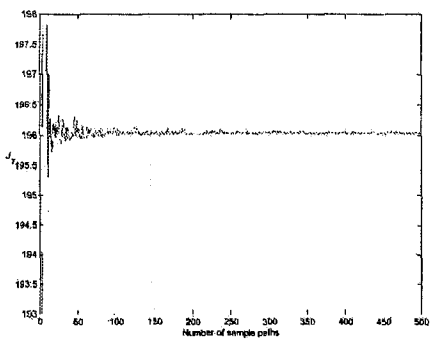
The objective is to find out a optimal control λ^* that minimizes the cost functional (5.6) for the interval I_j. Again we are interested in the expected individual costs given by

$$\overline{J}(\lambda) \equiv (1/k) E\Big\{ \sum_{j=1}^{k} (-J_T + J_C + J_L + J_S) \Big\} = -\overline{J}_T + \overline{J}_C + \overline{J}_T + \overline{J}_S \tag{5.11}$$

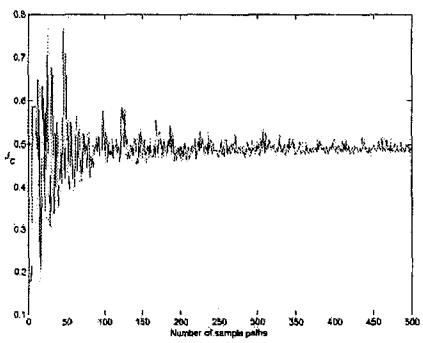
where $\overline{J}$, $\overline{J}_T$, $\overline{J}_C$, $\overline{J}_L$, $\overline{J}_S$ are the expected throughput cost, the expected congestion cost, the expected packet losses cost and the expected synchronization cost at one time interval respectively. Monte Carlo method has been used to calculate the expected cost of the real time system.



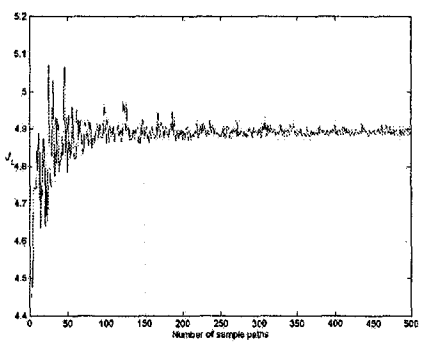
(a) Expected total cost



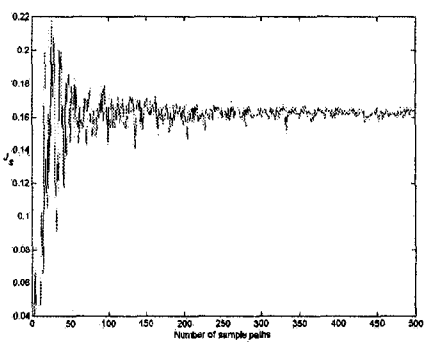
(b)Expected throughput cost



(c)Expected congestion cost



(d)Expected packet losses cost



(e)Expected synchronization cost

Figure 5.7: Individual average costs

5.6.2 Performance Comparison

We can see in figure 5.7 that a set of 300 traces of sample paths is good enough for the determination of individual expected costs. Table 5.4 shows these expected individual costs according to figure 5.7.

	$\overline{J}$	$\overline{J}_T$	$\overline{J}_C$	$\overline{J}_L$	$\overline{J}_S$
1 time interval delay	-190.45	196	0.49	4.9	0.16

Table 5.4: Individual expected costs

Compared with the costs shown in table 5.2,it can be seen that the feedback controller performs better than the real time controller for the same set of parameters. We analyze that the controls made by the real time system are optimal for each of time slots but might not for overall simulation time since the AQM scheme is controlled without the requirement of feedback information. In fact, in light of the time delay, the AQM scheme of the real time system usually detects congestion rather late and thereby could ask more than one source to cut window sizes at the same time in the case of congestion. This can result in a global synchronization of TCP flows (see figure 5.8), followed by a sustained period of lowered link utilization, reducing overall throughput.

On the other hand, the performance of the real time control can be accepted provided the feedback control law is rather difficult to apply under some conditions, especially in the real time process. In this case, the real time control algorithm can be considered as an alternative control mechanism to prevent network from congestion. We also notice that it takes much longer time for the feedback control to run optimization process (in order to get optimal parameters) before it is applied in AQM scheme, while the real time approach is a

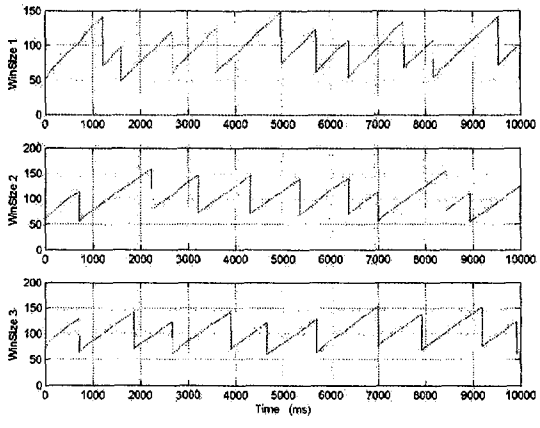


Figure 5.8: Variation of window sizes for real time control

simple control law that can be physically implemented immediately in TCP-RED dynamic model once the algorithm is well done.

Chapter 6

Conclusion

A variant of RED model presented in this thesis formally defines the interactions between TCP connections and the AQM scheme in computer networks.

The AQM scheme can be modeled as a feedback control system governed by a stochastic differential equations driven by doubly stochastic point processes with intensities being the controls for each of the TCP connections. The frequency of warning signals represented by the intensities are taken as functions of available information such as the router and buffer loads. The optimal (linear) feedback control law based on such information is obtained by Random Recursive Search (RRS) technique. According to the feedback control law proposed, the controller observes the behavior of all the queues (at buffers and the multiplexer) and controls every individual intensity by sending congestion signals (warnings) to the sources for adjustment of their transmission rates. The simulation results demonstrate that the model and the methodology proposed can improve the system performance significantly via maximizing the link utilization and minimizing congestion, packet losses and global synchronization.

However, we must admit that there is still a need to do further studies on sensitivity

analysis to validate the robustness of feedback control. In order to address these issues, we foresee following aspects to be studied:

- Sensitivity to the link capacity. Under this policy, C_i must adjust its bandwidth along with the variation of C_n.
- Sensitivity to the number of flows. In our work, we have fixed the number of connections. But in reality, the traffic load is changing all the time. It is a must to see if the control performance of the feedback controller is sensitive to the number of TCP flows.
- Sensitivity to the round trip time (RTT). An increase of RTT not only degrades the control performance of an AQM algorithm but also leads the system to fall into instable status. Thus, the effect of RTT should be taken into account in designing an AQM that is robust, especially in wide-area networks (WAN) environments.

In the thesis, we also examine the real time control law, by which the system computes the cost functional online and execute the optimal control in real time, based on the same RED model. We make some comparison with feedback control laws in terms of system performances. Despite the performance we still recommend the real time control can be considered as an alternative tool of feedback control because it has been developed without the optimization process.

Bibliography

[1] N. U. Ahmed and X.H. Ouyang, *Suboptimal RED Feedback Control for Buffered TCP Flow Dynamics in Computer Network*, to be appear.

[2] Jean Walrand and Pravin Varaya, *High Performance Communication Networks*, Morgan Kaufmann, second edition, 2000.

[3] B. Braden et al., *RFC 2309: Recommendations on queue management and congestion avoidance in the Internet*, , April 1998.

[4] S. Floyd and V. Jacobson, *Random early detection gateways for congetsion avoidance*, IEEE/ACM Transactions on Networking, 1: 397-413, August 1993.

[5] M. Christiansen, K. Jeffay, D. Ott, and F. Smith, *Tuning RED for Web Traffic*, In Proceedings of ACM SIGCOMM, Aug. 2000.

[6] C. Diot, G.Iannaccone, and M. May, *Aggregate Traffic Performance with Active Queue Management and Drop from Tail*, Sprint ATL Technical Report TR01-ATL-012501, 2001.

[7] Jae Chung and Mark Claypool, *Analysis of Active Queue Management*, 2nd IEEE International Symposium on Network Computing and Applications (NCA), April 2003

[8] N. U. Ahmed and Cheng Li, *Suboptimal Feedback Control of TCP Flows in Computer Network Using Random Early Discard (RED) Mechanism*, Mathematical Problems in Engineering, 2005, Vol.5, 477 - 489.

[9] S. Floyd, R. Gummadi, S. Shenker, and ICSI. *Adaptive RED: An algorithm for increasing the robustness of RED active queue management*, Berkeley, CA. [Online]http://www.icir.org/floyd/red.html

[10] Jay Aikat, Jasleen Kaur, F. Donelson Smith, Kevin Jeffay, *Variability in TCP Roundtrip Times* IMC03, October 2729, 2003, Miami Beach, Florida, USA.

[11] Yu.A.Shreiider, *Monte Carlo method: the method of statistical trials*, Oxford, New York, 1966

[12] A. Dubi, *Monte Carlo applications in systems engineering*, Chapter 3, John Wiley & Sons, 2000

[13] N.U.Ahmed, Cheng Li, *Stochastic Models for Traffic Dynamics and Access Control Mechanism in Computer Network*, Engineering Simulation, 2004

[14] Vegesna, Srinivas, *IP Quality of Service*, chap.6, Cisco press, 2001

[15] N.U.Ahmed and Cheng Li, *Optimum feedback strategy for access control mechanism modelled as stochastic differential equation in computer network*, Mathematical Problems in Engineering, 2004, Vol.3, 263-276

[16] T. Ye and S. Kalyanaraman, *A Recursive Random Search Algorithm For Large-Scale Network Parameter Configuration*, Technical report. ECSE Depamnent, Rensslacr Polytnhnique Institute, Des 2001.

[17] V. Guffens and G. Bastin, *Optimal Adaptive Feedback Control of a fluid Flow Network Buffer*, American Control Conference ACC2005, Portland, June, 2005, pp. 1835 - 1840.

[18] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations.* New York: Springer-Verlag, 1992.

[19] N.U. Ahmed, *Elements of finite-dimensional systems and control theory* , Harlow: Longman Scientific & Technical, 1988.

[20] Gene F. Franklin, J. David Powell, Abbas Emami-Naeini, *Feedback control of dynamic systems*, Upper Saddle River, NJ : Pearson Prentice Hall, c2006.

[21] Kuo, Benjamin C., *Automatic Control Systems (6th ed)*, New Jersey: Prentice Hall, 1991.

[22] Floyd, S., *RFC 2914: Congestion control principles*,2000.

[23] M. Allman, V. Paxson, and W. Stevens, *TCP Congestion Control*, IETF RFC 2581.

[24] V. Guffens and G. Bastin, *Optimal Adaptive Feedback Control of a fluid Flow Network Buffer*, American Control Conference ACC2005, Portland, June, 2005, pp. 1835 - 1840.

[25] Tomoya Eguchi, Hiroyuki Ohsaki, and Masayuki Murata, *On Control Parameters Tuning for Active Queue Management Mechanisms using Multivariate Analysis*, Proceedings of the 2003 Symposium on Applications and the Internet (SAINT03), 2003

[26] N.U.Ahmed, *Dynamic Systems and Control with Applications*, World Scientific Publishers, 2006

[27] Padhye, J.S. Floyd, *On Inferring TCP Behavior*, Computer Communication Review ACM-SIGCOMM, Vol.31, Augest 2001.

Appendix: Source Codes of The Simulation Programs

% Calculating expect J value to find a best set of control parameters to get min J,

% This objective Function differ from cost function at the dropping point not start at lambda=0;

% Add feedback in router side

$clear;\quad tic;$

$wn = 3;$ %wn is number of TCP windows

$C = 2000;\quad beta = 2;\quad c = beta * C/wn;$

$c(1) = c;\quad c(2) = c;\quad c(3) = c;$

$num = 10000;$ %number of time interval used

$Q = 40;\quad q = 0.3 * Q * ones(wn, num);\quad av_q = 0.3 * Q * ones(wn, 1);$

$Q_M = 200;\quad q_m(1) = 0.3 * Q_M;$ % main queue size

$alph = 0.02;\quad dt = 0.01;$ %time slot

$W = ones(wn, 1);\quad W(1) = 0;\quad W(2) = 0;\quad W(3) = 0;$ %initial windows to 0

$Win_size = zeros(wn, num);\quad Win_rate = zeros(wn, 1);\quad tt = zeros(wn, 1);$

$R = zeros(num, wn);$ %Congestion indications– 0 or 1

$I = zeros(wn, 1);$ %output of sub-queues

$num_delay = 1;$ %How many time interval delay

$tt = zeros(wn, 1);$ %initial windows to 0

$lam = zeros(num, wn);$ $la = zeros(wn, 1);$ $lam1 = 100;$ $lam2 = 80;$

$lam3 = 80;$ $lam4 = 80;$ $test = 1;$

$while$ $test < 301,$ %run monte carlo

$cc1 = 0;$ %the number of over 0.9*Q_M ; counter of queue overflowed

$cc3 = 0;$ %the number of $q_m = 0$

$q_loss1 = 0; q_loss2 = 0; q_loss = 0;$ %initial queue loss

$sign_sum = 0;$

for $i = 1 : (num - 1)$

$sign = 0; cc2 = 0;$ %the number of q=0

for $ff = 1 : wn$

$delay = i - num_delay;$

if $delay < 1$

$delay = 1;$

end

$[Win, Win_rate, T_rtt, d_w]=$

$TCP_windows(c(ff), C, q(ff,i), q_m(i), W(ff), tt(ff), R(delay, ff), dt);$

$W(ff) = Win;$

$Win_size(ff,i) = Win;$

$tt(ff) = T_rtt;$

$W_Rate(ff,i) = Win_rate;$

$if d_w < 0$

$sign = sign + 1;$

end

$if \quad q(ff,i) == 0$

$I(ff) = min(c(ff), W_Rate(ff, delay));$

$cc2 = cc2 + 1;$

$else$

$I(ff) = c(ff);$

end

$dq = (-c(ff) + W_Rate(ff, delay)) * dt;$

$q(ff, i+1) = max(0, q(ff,i) + dq);$

$if q(ff, i+1) > Q$ %avoid queue length is larger than Queue size

$q_loss1 = q_loss1 + q(ff, i+1) - Q;$ %cause queue loss here

$q(ff, i+1) = Q;$

end

end % end of for

$if \;\; sign > 1$
$sign_sum = sign_sum + 1;$
end

$dq_m = (-C + sum(I)) * dt;$
$q_m(i+1) = max(0, q_m(i) + dq_m);$ %main queue

$if \;\; q_m(i+1) == 0$
$cc3 = cc3 + 1;$
end

$if \;\; q_m(i+1) > 0.9 * Q_M$
$cc1 = cc1 + 1;$
end

$if \;\; q_m(i+1) > Q_M$ %avoid queue length is larger than Queue size
$q_loss2 = q_loss2 + q_m(i+1) - Q_M;$ %cause queue loss here
$q_m(i+1) = Q_M;$
end

%RED algorithm
$la(i,1) = (pp1(1) * q_m(i+1)) + (pp1(2) * q(1, i+1)) + pp1(3);$
%lamda values as a function of average q

$la(i,2) = (pp2(1) * q_m(i+1)) + (pp2(2) * q(2,i+1)) + pp2(3);$
$la(i,3) = (pp3(1) * q_m(i+1)) + (pp3(2) * q(3,i+1)) + pp3(3);$

$for \quad ff = 1 : wn$
$lam(i+1,ff) = max(0, la(i,ff));$
$R(i+1,ff) = poissrnd(lam(i+1,ff),1,1);$

$if \quad R(i+1,ff) > 1$
$R(i+1,ff) = 1;$
end

end

end %end of num

$throughput(test) = -C * (num - cc3)/1000;$
$congestion(test) = cc1;$
$loss(test) = q_loss1 + q_loss2;$
$syn(test) = sign_sum;$
$new_cost(test) = ((-C*(num-cc3)*lam1)/1000+(cc1*lam2)+((q_loss1+q_loss2)*lam3)+$
$(sign_sum * lam4))/num;$
$test = test + 1;$

end %end of test while

$f = mean(new_cost)$

$Throughput = mean(throughput)$
$Congestion = mean(congestion)$
$Loss = mean(loss) Syn = mean(syn)$

% TCP/RED Different number of users
% polynomial function with ave q size, optimize with RRS
% Use feval function which can replace cost function easily

$n = 44$; %Initialize exploration parameters p, r,
$n = ln(1-p)/ln(1-r) r = 0.1$;
$L = 22$; %Initialize exploitation parameters q, v, st, L=ln(1-q)/ln(1-v);
$st = 0.1$; $Range2 = -2$; $Range1 = 2$; $test = 0$; $f_x 0 = inf$; $initial_cost = inf$;
$jj = 1$; %index of F
$sk = 0.5$; %shrink search areas
$ord = 3$; %number of polynamial coeficients
$stop_counter = 0$;
$cost_fun = 'cost'$; $counter = 0$;
$rr = 0$; %initial index for x axis, y axis

$while \quad test < 44$, %Take n random samples from parameter space D;
$pp = random('unif', Range2, Range1, ord, 1)$;
$rr = rr + 1$;
$new_cost = feval(cost_fun, pp)$;

$if \;\; ((f_x0 - new_cost) > 0)$

$f_x0 = new_cost;$ % x0 = $arg_min(f(xi))$, y_r =f(x0), add f(x0) to the threshold set F;

$x0 = pp;$

$s = struct('J', f_x0, 'No_function_call', counter);$

$disp(s);$

end

$test = test + 1;$

$counter = counter + 1;$

end

$y_r = f_x0; \quad f_opt = f_x0; \quad F(1) = f_x0; \quad x_opt = x0; \quad x_c = x0; \quad i = 0;$

$exploit_flag = 1;$

$while \;\; stop_counter < 300,$ % stopping criterion is not satisfied do

$if \;\; exploit_flag == 1$ % then Exploit flag is set, start exploitation process

$Range3 = Range1;$

$Range4 = Range2;$

$j = 0;$

$f_c = f_x0;$

$x1 = x0;$

$while \;\; Range3 > st,$ % 2nd while loop

$x2 = x_c + random('unif', Range4, Range3, ord, 1);$

% Take a random sample x0 from ND;

$f_x1 = feval(cost_fun, x2);$

$rr = rr + 1;$

$cost_j(rr) = f_x1;$

$if \quad f_x1 < f_c$ %Find a better point, re-align the center of sample space to the new point

$x_c = x2;$

$f_c = f_x1;$

$j = 0;$

$s = struct('J', f_c, 'No_function_call', counter);$

$disp(s);$

$else$

$j = j + 1;$

end

$if \quad j == LRange3 = sk * Range3;$

$Range4 = sk * Range4;$

$j = 0;$

end

end %end 2nd while

$exploit_flag = 0;$

$if \quad f_c < f_opt f_opt = f_c;$

$x_opt = x_c;$
end

$counter = counter + 1;$

end % end if of exploit process

x2=random('unif',Range2,Range1,ord,1); %Take a random sample x0 from S;
$counter = counter + 1;$
$f_x0 = feval(cost_fun, x2);$
$rr = rr + 1;$
$stop_counter = stop_counter + 1;$

$if \ \ f_x0 < y_rexploit_flag = 1;$
$x_c = x2;$
end

$if \ \ f_x0 < initial_cost$
$initial_cost = f_x0;$
end

$if \ \ i == n$ % Update the exploitation threshold every n samples in the parameter space
$jj = jj + 1;$
$F(jj) = initial_cost;$ % Add min(f(xi)) to the threshold set F;
$y_r = mean(F);$

$i = 0;$

$initial_cost = inf;$

end

$i = i + 1;$

end %end 1st while

$toc;$

%real time control

$clc;$ $clear;$ $tic;$

$wn = 3;$ %wn is number if TCP windows

$C = 2000;$

$num = 10000;$

$Q = 200;$

$q(1) = 0.3 * Q;$

$av_q(1) = 0.4 * Q;$

$alph = 0.02;$

$dt = 0.01;$

$W = ones(wn, 1); W(1) = 0; W(2) = 0; W(3) = 0;$

$Win_size = zeros(wn, num);$

$Win_rate = zeros(wn, 1);$

$R = zeros(wn, num);$ $R1 = zeros(wn, num);$

$num_delay = 1;$ %How many time interval delay $tt = zeros(wn, 1);$

$lam = zeros(num, wn);$ $la = zeros(wn, 1);$

$lam1 = 100;$ $lam2 = 80; lam3 = 80;$ $lam4 = 80;$

$test = 1;$

$SUMCOST = zeros(1, 501);$

$while$ $test < 501,$ %run monte carlo

for $i = 1 : (num - 1)$

$cc3 = 1;$ % thoughput

$cc2 = 0;$ %Synchronization

$cc1 = 0;$

% the number of over 0.9*Q_M ; counter ofqueue overflowed

$q_loss = 0;$ %initial queue loss

$delay = i - num_delay;$ %i-round(mean(tt)/dt/2);

if $delay < 1$

$delay = 1;$

end

for $ff = 1 : wn$

$[Win, Win_rate] = TCP_windows1(C, q(i), W(ff), R(ff, delay), dt);$

$W(ff) = Win;$

$Win_size(ff, i) = Win;$

$W_Rate(ff) = Win_rate;$
end

$if \;\; q(i) == 0$
$dq = (sum(W_Rate) - min(C, sum(W_Rate))) * dt;$
$cc3 = 0;$
$else$
$dq = (-C + sum(W_Rate)) * dt;$
end

$q(i+1) = max(0, q(i) + dq);$

$if \;\; q(i+1) >= 0.9 * Q$

$for \;\; ff = 1 : wn$

$if \;\; W_Rate(ff) == max(W_Rate)$

$R(ff, i+1) = 1;$
$R1(ff, i+1) = 1;$

$if \;\; q(i) >= 0.9 * Q$
$if \;\; R(ff, i) == 0$
$cc2 = 1;$
end

end

$else$

$R(ff, i+1) = 0;$
$R1(ff, i+1) = 0;$
end

end

$cc1 = 1;$
$q_loss = q(i+1) - 0.9 * Q;$
end

$if \quad q(i+1) > Q$
$q(i+1) = Q;$
end

$throughput(i) = 100 * C * cc3/1000;$
$congestion(i) = 80 * cc1;$
$loss(i) = 80 * q_loss;$
$syn(i) = 80 * cc2;$
$cost(i) = -throughput(i) + congestion(i) + loss(i) + syn(i);$
end

$TotalCost(test) = sum(cost)/num + 10;$

$SUMCOST(test) = (sum(SUMCOST) + TotalCost(test))/test;$

$test = test + 1;$

end

$plot(SUMCOST(1:500));$

$xlabel('Numberofsamplepaths');$

$ylabel('J','FontSize',12,'FontAngle','italic','Rotation',0);$

$toc;$

%TCP window Function

$function[Win, Rate] =$

$TCP_window(C, q, WWW, RRR, dt)R1 = random('unif', 0.01, 0.2) + q/C;$

$dw = (1/R1) * dt - 0.5 * WWW * RRR;$

$Win = max(0, (WWW + dw));$

$Rate = Win/R1;$