PERFORMANCE ANALYSIS OF A MODIFIED NON-LINEAR RED ALGORITHM FOR CONGESTION AVOIDANCE IN TCP/IP NETWORKS

Jiang Chang

A thesis
submitted to the Faculty of Graduate Studies
in partial fulfilment of the requirements
for the degree of

Master of Science

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba
October 2004

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

PERFORMANCE ANALYSIS OF A MODIFIED NON-LINEAR RED ALGORITHM FOR CONGESTION AVOIDANCE IN TCP/IP NETWORKS

BY

JIANG CHANG

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

Random Early Detection (RED) algorithm [FJ93] has been introduced as a congestion avoidance scheme for the Internet. As a result, there have been considerable research efforts in studying the performance of RED and various modifications to the classical RED algorithm have been proposed.

However, previous studies and modifications have often focused on adaptively tuning RED parameters. The weakness of RED in avoiding global synchronization of TCP flows are not thoroughly addressed. Although May et al. [MBB00] have investigated and analyzed this weakness and remarked that the number of consecutive packet drops is higher with RED than Tail Drop, their conclusion drawn is based on RED linear dropping model (geometric dropping scheme). Floyd et al. [FJ93] have already shown that the RED non-linear dropping model (uniform dropping scheme) yields better performance in terms of avoiding TCP synchronization than the linear dropping model. This thesis examines and analyzes the synchronization of TCP flows by extending May's analytic model used for linear RED dropping algorithm to non-linear RED dropping algorithm. Based on the analysis of these two models, we propose a modified non-linear RED dropping algorithm to avoid synchronization of TCP flows. The results

of mathematical analysis show that our modified non-linear RED algorithm has much lower consecutive dropping probability than classical linear (May's analysis) and non-linear RED algorithms, even though it brings a little bit higher queue delay than classical linear and nonlinear RED algorithms. It means that our scheme can avoid the synchronization of TCP flows more efficiently.

Acknowledgements

I would like to express my heartfelt gratitude to my advisor, Professor Attahiru S. Alfa for his genuinely enthusiastic guidance and valuable comments throughout the year.

I would like to thank my thesis committee members Dr. Ken Ferens and Dr. Rasit Eskicioglu for their time, valuable suggestions and comments.

I would like to thank my fellow students and friends for their companionship.

I would like to express my gratitude to my wife and my parents for all the support and encouragement they have provided during the work.

Table of Contents

List of Tables

List of Figures

1 Introduction

This chapter describes the general concepts and issues of Internet congestion control and Random Early Detection (RED), and presents the motivation and goal of my research. The thesis structure is summarized as well.

1.1 Motivation and Goal

Traffic volumes in the Internet has been growing at a rapidly increasing rate in the past few years. This growth in traffic creates great challenges for controlling and managing the traffic flows in the network. In such a situation, one of the most important issues is congestion control, i.e., how the network should react in situations where the traffic load becomes too high, threatening the stable operation of the network. In the Internet, this problem has been traditionally handled by the use of the Transmission Control Protocol (TCP), which is a reliable packet transfer protocol aiming at providing each TCP flow a fair share of the limited bandwidth. In the current Internet a vast majority of the data (e.g., HTML, FTP, e-mail, and telnet traffic) is transmitted by using TCP. However, it has been discovered that the use of TCP is not enough to guarantee the stable operation of the network and that TCP alone does not always

achieve a fair sharing of the bandwidth. In addition, the amount of traffic generated by non-responsive sources, e.g., UDP sources, has been steadily rising with the increased use of the Internet for transmitting real time traffic streams, such as real-time audio and video. In general, non-responsive flows do not react to congestion by cutting down their sending rates and can thus obtain more than their fair share of the bandwidth at the expense of the responsive TCP flows.

The traditional technique for managing router queue length is to set a maximum length (in terms of packets) for each queue, accept packets for the queue until the maximum length is reached, then drop subsequent incoming packets until the queue length decreases because a packet from the queue has been transmitted. This technique is known as Tail Drop. This method has served the Internet well for many years, but it has two serious drawbacks which are lockout and Global Synchronization. The lockout in some situations allows a single connection or a few flows to monopolize queue space, preventing other connections from getting room in the queue. This lockout phenomenon [BCC98] is often the result of synchronization or other timing effects. The Tail Drop discipline allows queues to maintain a full status for long periods of time, since it signals congestion only when the queue has become full. When multiple TCP sessions sharing a common gateway arrive at a full buffer, the impact is (typically) simultaneous packet loss and reduction of transmission rate in all flows, resulting in an oscillatory and bursty behavior termed global synchronization reported experimentally by [ZC90]. This leads to link capacity under-utilization and exacerbates the inherent bias of TCP congestion control algorithm against higher delay flows.

For these reasons, the Internet Engineering Task Force (IETF) has recommended in their standards the use of new congestion control mechanisms, so called Active Queue Management (AQM) methods that are implemented in the buffers of the routers of the network [BCC98]. Basically, these mechanisms attempt to reduce the unfairness between responsive and unresponsive flows, avoid global TCP synchronization and inhibit the build up of congestion in the buffers by implicitly signaling the responsive traffic sources to reduce their sending rates before the buffer becomes overloaded. One of the most prominent AQM methods is the Random Early Detection (RED) algorithm proposed by Floyd and Jacobson in [FJ93]. The algorithm has also been implemented in commercially available routers. It has been designed to work in cooperation with the TCP sources, and it has been shown [FJ93] by using simulations that the algorithm is able to alleviate the problem of global synchronization of the TCP sources, which may happen when a congested buffer overflows, as well as to increase fairness between high speed and low speed TCP flows. RED drops packets before the actual physical queue is full. It operates based on an average queue length that is calculated using an exponential weighted average of the instantaneous queue length. RED drops packets with certain probability depending on the average length of the queue. The drop probability increases from zero to a maximum drop probability as average queue size increases from a minimum threshold to the maximum threshold. If the average queue size goes above the maximum threshold, all packets are dropped.

Since the introduction of RED, many researchers have carried out investigations about the behaviors and performances of RED, and proposed a variety of enhancements and

changes to router management to improve congestion control. However, previous studies and modifications have often focused on adaptively tuning RED parameters. The weakness of RED in avoiding global synchronization of TCP flows are not thoroughly studied. Although May et al. have investigated and analyzed this weakness and remarked that the number of consecutive packet drops higher with RED than Tail-Drop and concluded that deploying RED does not alleviate the synchronization of TCP flows and in fact might contribute to the global TCP synchronization, their conclusion drawn are based on linear dropping model (geometric dropping scheme). S. Floyd and V. Jacobson [FJ93] have already shown that the non-linear dropping model (uniform dropping scheme) yields better performance in terms of avoiding TCP synchronization than the linear dropping model. Hence, the conclusion made by May et al. [MBB00] may not be valid. One of the main reasons that the RED non-linear dropping model is considered undesirable is that it brings more difficulty in performing the mathematical analysis.

The goal of this project is to investigate May's claim regarding synchronization of TCP flows by extending the analytic model used for linear RED dropping algorithm to non-linear RED dropping algorithm. We examine and analyze the difference between these two schemes. Furthermore, we propose a modified non-linear RED dropping algorithm to reduce the likelihood of synchronization of TCP flows. We introduce a latency factor and a compensation factor for non-linear RED dropping model to make RED gateway to delay dropping packet until at least some packets have been accepted by RED gateway in order to get a mandatory separation between consecutive packet drops to avoid

global synchronization of TCP flows. Then we compare and analyze the performance of our modified non-linear RED algorithm with that of classical linear RED model (May's model) and nonlinear RED model by the mathematical analysis. To obtain the performance measures of interest, two technics exist: simulate the system or solve the model mathematically. Our research concentrates on the latter. We derived a series of closed-form equations to solve our analysis model.

1.2 Structure of Thesis

This thesis is organized as follows. Chapter 2 presents the background of congestion control mechanisms in TCP/IP network and active queue management, and then gives a literature review on RED research. Chapter 3 describes the RED algorithms including RED design goals, linear RED dropping model (geometric dropping model) and non-linear RED dropping model (uniform dropping model), and then give the discussion on these two models [FJ93]. In chapter 4 we develop a modified non-linear RED algorithm based on the analysis of RED algorithms in Chapter 3. We give problem statement and then introduce latency and compensation factors to improve the performance of consecutive packet drops in RED gateway. We also verify our modified non-linear dropping model to be uniform dropping model because [FJ93] shows that uniform dropping model has better performance in alleviating the synchronization of TCP flows. Chapter 5 gives our mathematic analysis for non-linear RED and modified non-linear RED algorithms. We derive a series of closed-form equations which are

the stationary distribution of observed queue occupancy, stationary distribution of the number of packets between consecutive drops, stationary dropping probabilities, the expectation and variance of the number of consecutive packet drops and the correlation between stationary average queue occupancy and the offered load based on our approximations. Finally, we give a closed-form solution to set the latency and compensation factors for our modified non-linear RED algorithm. In chapter 6, we analyze and compare the performance of different RED dropping models using the analysis models presented in chapter 5. The various numerical scenarios are investigated for comparing the performance of linear RED, non-linear RED and the modified non-linear RED algorithms in the characteristics of consecutive packet drops. The conclusions of this thesis and the future work are presented in Chapter 7.

2 Background and Literature Review

This chapter gives an overview of congestion control and management mechanisms used in TCP/IP networks, and then describes the background of active queue management algorithms proposed, as well as the literature review on the RED research.

2.1 TCP Congestion Control Mechanisms

It is important to avoid high packet loss rates in the Internet. When a packet is dropped before it reaches its destination, all of the resources it has consumed in transit are wasted. During the mid 1980s, the Internet meltdown phenomenon was first observed, which is also called congestion collapse [Jac88]. Originally, TCP included window based flow control mechanism as a means for the receiver to control the amount of data sent by a sender. The flow control mechanism was used to prevent overflow of the receiver's data buffer space available for the TCP connection. In 1986, in order to fix Internet meltdown, Jacobson developed the congestion avoidance mechanisms which are now used in TCP implementations. These mechanisms operate in the end-hosts to cause TCP connections to back off during congestion. Those TCP flows are said to be responsive to congestion signals (i.e., packet loss) from the network. It is these

TCP congestion avoidance algorithms that are still being used to prevent the congestion collapse of today's Internet. TCP congestion control is window-based. The sender keeps a congestion window (CWND) whose size limits the number of unacknowledged packets the sender can send in the network. Upon receiving acknowledgments for successfully transmitted data, the sender increases its transmission rate by incrementing the size of its congestion window. At some point in time, the rate at which TCP sends its packets eventually exceeds the network's capacity to deliver them. When this happens, queues build up in the network routers and overflow, causing packets to be dropped. TCP assumes that all packet loss is due to congestion and reduces its congestion window upon detecting a loss. TCPs congestion control algorithm is fairly straightforward. When a connection starts up, it attempts to ramp up its sending rate quickly by exponentially increasing its congestion window until it reaches an implementation specific value (SSTHRESH). This stage is called slow-start and allows the source to double its congestion window, and thus its sending rate, every round-trip time. In order to prevent excessive losses due to an exponentially-increasing sending rate, TCP senders typically employ what is known as the congestion-avoidance algorithm showed in [Jac88] and [Ste97], a modification to TCP first deployed in Reno variants of TCP. In this algorithm, TCP uses the SSTHRESH value to approximate the window size which the network can support. When the window size exceeds this threshold, TCP enters the congestion avoidance phase. In this phase, the window is increased at a much slower rate of one segment per round-trip time. When the offered load increases above network capacity, packets are eventually dropped. One way in which TCP detects a

packet loss is through the receipt of a number of duplicate cumulative acknowledgments from the receiver [Jco90]. Instead of waiting for TCP timeout, when duplicate acks are received for an earlier packet, TCP infers that a packet loss has occurred and immediately reduces its sending rate in half by halving its congestion window and sets SSTHRESH to the new value of the congestion window. These mechanisms are called fast retransmit and fast recovery (instead of entering slow start, congestion avoidance is entered.

When congestion is severe enough such that packet loss cannot be inferred in such a manner, TCP relies on a separate, retransmission timeout mechanism to trigger subsequent retransmissions of lost packets. When a retransmission timeout occurs, TCP reduces its window size to one segment and retransmits the lost segment. To prevent continual retransmissions in times of severe congestion and network outages, TCP employs an exponential back-off algorithm. In particular, if the sender continually sends the same segment, but receives no acknowledgments for it, TCP doubles its retransmission timeout interval. Upon receipt of an acknowledgment for subsequent new segment, TCP resets the timeout interval and resumes its normal sending. Figure 2.1 shows a graphical picture of how TCP slow-start and congestion avoidance work.

As the figure shows, TCP initially starts with a congestion window of 1. The window is then doubled every round-trip time. When the congestion window reaches SSTHRESH, TCP slows its rate of increase. Eventually, when the transmission rate of the connection overwhelms the bottleneck link, packets are dropped. This loss is detected by TCP which then reacts by halving the congestion window (assuming the

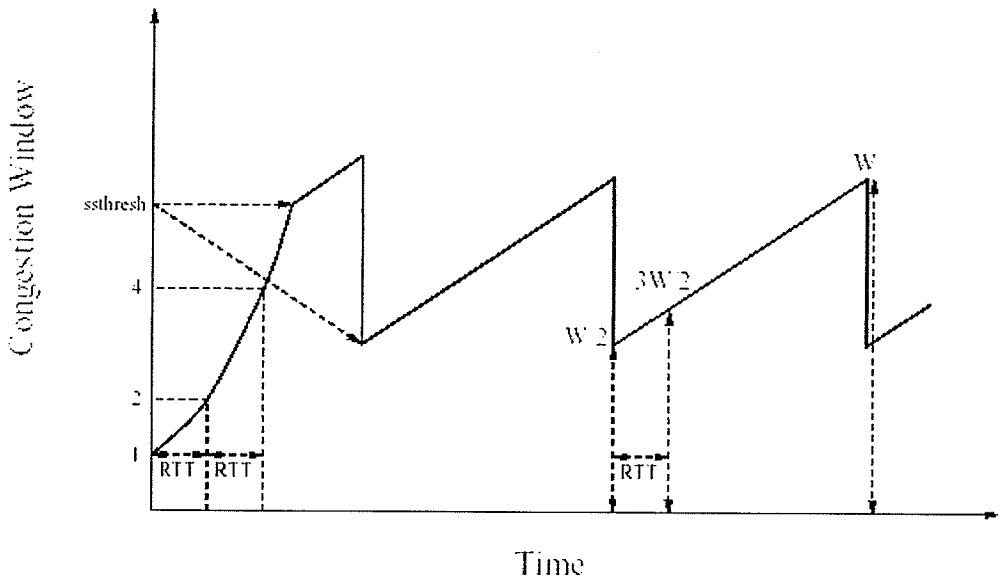

Figure 2.1: An example of how TCP slow-start and congestion avoidance works

fast-retransmit and fast-recovery mechanisms are triggered). As the figure shows, upon recovering from congestion, the TCP sender enters the congestion avoidance phase in which the window is increased linearly at a rate of one segment per round trip time. In steady state, TCP then oscillates between a window of W and $W/2$ where W depends on the capacity of the network and the number of connections currently active over the bottleneck link.

Given the importance of TCP and its congestion control mechanisms to the health of the Internet, there have been a number of proposed modifications to its algorithms. One modification which has been proposed is selective acknowledgments(SACK)[MMFR96]. SACK augments TCPs cumulative acknowledgment mechanism with additional information that allows the receiver to inform the sender which segments it is missing. By

specifying this information, the TCP sender can make more intelligent decisions in determining when packets have been lost and in identifying which segments should be retransmitted. This helps TCP detect congestive loss more quickly and eliminates unnecessary retransmissions by TCP senders. Another set of proposed TCP modifications focuses on congestion recovery. TCP is ACK-clocked, often sending only after it has received acknowledgments for previously transmitted packets. When there are insufficient packets or acknowledgments in flight to trigger TCP sends, a retransmission timeout must occur before the TCP source can resume sending. Because the Reno variant of TCP freezes its window while recovering from congestion, it often induces a subsequent retransmission timeout since the source does not send packets upon receiving acknowledgments in the recovery phase. To address this problem, a simple observation is made. When a TCP sender receives any type of acknowledgment, it is a signal that a packet has left the network and should thus allow the TCP sender to inject an additional packet without causing further congestion. This modification allows TCP to maintain its ACK-clocking and prevents unnecessary retransmission timeouts. Both the FACK [MM96] and New Reno [FH99] and [Hoe96] modifications use this observation to improve TCP performance. Finally, more radical changes to TCPs congestion control algorithms have been proposed. In current incarnations of TCP, the congestion window follows a sawtooth-like pattern where the congestion window is continually increased until packet loss occurs. While this allows TCP to probe for additional bandwidth, such behavior eventually induces packet loss. The idea behind the Tri-S [WC91] and [WC92] and Vegas [BOP94] modifications is to change the

congestion avoidance phase so that it only performs its linear increase when the network is not congested. In both algorithms, if the round-trip times indicate an increase in delay due to queues being built up in the network, the TCP source either decreases or fixes the size of the congestion window rather than increasing it. It's reported that they gives better throughput than Reno.

2.2 Active Queue Management

The congestion control mechanism of TCP was designed to work without any knowledge on underlying network including the router's algorithm. Namely, the congestion control mechanism of TCP assumes nothing about a gateway's operation. It is because neither a packet scheduling discipline nor a packet discarding algorithm of the gateway is known by the source host in real networks. The sending sources reduce their transmission rates only after detecting packet loss. This is a problem since a considerable amount of time may pass between when the packet is dropped at the router and when the source actually detects the loss. In the meantime, a large number of packets may be dropped as sources continue to transmit at a rate that the network cannot support. Because of these, the IETF has advocated the active queue management as a means to prevent packet loss.

Active Queue Management (AQM) denotes a class of algorithms designed to provide improved queueing mechanisms for routers. These schemes are called active because they dynamically signal congestion to sources; either explicitly, by marking packets (e.g.

Explicit Congestion Notification [Flo94]) or implicitly, by dropping packets [FJ93]. This is in contrast to Tail Drop queueing which is passive: packets are dropped if, and only if, the queue is full. The Internet Engineering Task Force (IETF) recommended the deployment of AQM in Internet routers in 1998. The main motivations given were the improvement of performance and the prevention of congestion collapse which may arise from the growth of non-responsive traffic on the Internet. The most well known and widely deployed active queue management mechanism is Random Early Detection (RED).

The main goal of RED is to provide better feedback to responsive flows. This goal has several parts. First, RED seeks to do a better job by detecting the onset of congestion instead of waiting until congestion is persistent and the queue is overflowing. Second, RED seeks to distribute feedback more evenly across all flows. Instead of the disparity in drops that can lead to the lock-out phenomenon observed with tail drop, RED seeks to insure that all flows have an equal percentage of their packets dropped. Finally, RED seeks to maintain shorter average queue occupancy. The intent is to avoid full queues and have queue space available to accommodate bursty arrivals, even during periods of modest congestion. RED accomplishes these goals by actively monitoring and managing the queue. Since a queue builds up when the offered load exceeds the links capacity and the queue drains when the load is less than the links capacity, a measure of the queues length over time is a good indicator of the state of congestion. Using the instantaneous queue occupancy is problematic as it may be the result of recent bursty arrivals and not persistent congestion. Thus, instead of using

instantaneous queue occupancy as a congestion indicator, RED uses the recent average queue occupancy. The RED algorithm maintains a running weighted average of queue occupancy. When the average is below a minimum threshold value, packets are simply enqueued and forwarded. When the average exceeds the minimum threshold, arriving packets are randomly dropped. The probability the arriving packet will be dropped is a function of the average queue occupancy and the number of packets that have arrived since the last packet was dropped. The probabilistic component of this mechanism allows RED to distribute the drops more evenly across all flows. Better distributed drops mean better feedback to all flows, allowing them all to back-off equally. Finally, dropping packets before the queue fills helps to maintain a shorter average queue. As a result, RFC2309 recommends that active queue management, specifically RED, be widely deployed in routers [BCC98].

2.3 Literature Review on RED

The introduction of RED has stirred considerable research interest in understanding its fundamental mechanisms. Primarily the methods used have been simulation based and have been aimed at solving some of the deficiencies of the original RED algorithm. Feng et al. [FKSS99b] propose an adaptive discarding mechanism where the maximum discarding probability parameter of RED is varied according to the number of flows present in the system. Floyd et al. [FGS01] make several algorithmic modifications to this proposal, while leaving the basic idea intact, and then evaluate its performance

using simulation. This revised version of Adaptive RED, which can be implemented as a simple extension within RED routers, removes the sensitivity to parameters that affect RED's performance and can reliably achieve a specific target average queue length in a wide variety of traffic scenarios. Lin and Morris [LM97] showed that RED is not efficient in the presence of non-adaptive (or "non-TCP-friendly") flows, such as UDP flows, and for that they propose a per flow version of RED (FRED). The problem of isolating misbehaving and well-behaved flows and providing specialized treatment for such circumstances has resulted, besides FRED, in a number of algorithms: RED+ [ZFH99], SRED [OLW99], BRED [AT99], REM [ALL00] and stochastic fair BLUE [FKSS01]. RED+ adds the functionality of identifying and discriminating high-bandwidth, unresponsive best-effort flows to RED gateways. SRED presents a mechanism for statistically estimating the number of active flows in a bottleneck link. Their mechanism is based on the idea of comparing the flow identifier of incoming packets to those from a randomly chosen zombie in a zombie list that records information regarding flows that have recently sent packets to the link. A hit is said to occur when the comparison is successful. The number of active flows can be estimated from the average hit rate. The method does not require keeping per-flow state. BRED maintains minimal flow state information and drops packet preventively, in an attempt to actively penalize the non-adaptive traffic that attempts to "steal" buffer space and therefore bandwidth from the active traffic flows. It tries to achieve a more balanced bandwidth allocation among the different flows. REM prescribes a way to control rate-adaptive flows to achieve social optimality. It provides each source with a congestion measure that is ag-

gregated over its path and can work with other source algorithms that can exploit this path congestion measure. Stochastic fair BLUE uses a technique to enforce fairness among a large number of flows. It scalably detects and rate-limits non-responsive flows through the use of a marking probability derived from the BLUE [FKSS99a] queue management algorithm and a Bloom filter. Clark and Fang [CF98] have suggested the use of another variant of RED, called RIO, to provide different classes of services in the Differentiated Services framework.

Analytical approaches have been also used for the performance analysis of RED controlled buffer. May et al. [MBB00] develop a analytic model of the RED algorithm in which they uses continuous-time Markov chain analysis to establish the transition rates, and consequently the stationary probability used to find the dropping probability. Peeters and Blondia [PB99] investigate RED in the presence of feedback traffic by a discrete-time analysis where a source consists of a 3-D Markov model. Sharma et al. in [SVL00] analyze the RED algorithm by ordinary differential equation (ODE) and develop asymptotic approximations for easing the burden of computing the performance indices of interest. By using a similar approach as in [SVL00], the performance of a RIO controlled queue has been studied by Kuusela and Virtamo [KV00]. Kohler et al. [KMV01] present a discrete time Markovian model for the TCP-RED interaction. A fixed point model for a network of AQM routers is given by Bu and Towsley [BT01]. V. Misra et al. [MGT00] discuss oscillations with RED. In RED buffer model, [MGT00] assumes an infinite and non-empty buffer and the model derivation results in an additional modeling parameter to be tuned. The stability analysis of the model

from [MGT00] is given by Hollot et al. in [HMGT01]. The authors use control theory and make a simplified linearization of the original system and are able to give necessary conditions for the system to be asymptotically stable. In [HMGT01], the authors have also presented an alternative AQM mechanism to replace RED, which has a better convergence speed than RED, by using a so called PI controller instead of a low pass filter, which the RED algorithm essentially is.

In summary, since the introduction of RED, many researchers have investigated the behavior and performance of RED, and proposed a variety of enhancements and changes to router management to improve congestion control. However, previous studies and modifications have often focused on adaptively tuning RED parameters. The weakness of RED in avoiding global synchronization of TCP flows are not thoroughly studied. We analyze and improve a specific feature of random early detection algorithms in alleviating global synchronization of TCP flows rather than tuning RED parameters themselves.

3 The RED and Modified RED Algorithms

In this chapter, first we detail RED algorithms and then give the problem description. Based on analysis of RED algorithms and its dropping models, we propose a modified RED algorithm.

3.1 The RED Algorithms

3.1.1 RED Design Goals

Random Early Detection mechanism for congestion avoidance in packet-switched networks was first introduced by Floyd and Jacobson [FJ93]. The purpose of the RED mechanism is to accompany TCP congestion control mechanism in order to avoid global synchronization of the TCP connections. The RED routers detect incipient congestion by keeping track of the average queue size. The RED router then implicitly notifies the traffic sources of congestion by dropping packets. When the average queue length exceeds a preset threshold, the router drops each arriving packet with certain probability. [FJ93] lists the following design goals for RED:

- Congestion avoidance

RED allows for queue congestion to be managed before a critical overflow point is reached. Also, keeping the queue size lower decreases delay for those packets that are not dropped.

- Global TCP synchronization avoidance

  By random early dropping packets, the number of consecutive drops can be reduced. Many Internet designers were concerned that consecutive drops when queues became full could cause global instability in the network as many queues signal their source to reduce their window at the same time.

- Fairness

  The RED gateway has no bias against bursty traffic. Namely RED will avoid a situation in which bursty traffic faces extreme packet loss compared to smooth traffic.

- Bound on average queue length

  RED should be able to control the average queue size and therefore control the average delay.

3.1.2 RED Algorithms

The RED algorithm consists of two main elements. The first element is the computation of the average queue length and second one is the packet dropping algorithms based on the average queue length and the number of packets that already have been accepted since the last dropped packet.

3.1.2.1 Computation of average queue length

The RED router computes the average queue length $\overline{q}_k$ using an exponential weighted moving average. After each packet arrival the value of the $\overline{q}_k$ is updated according to

$$\overline{q}_{k+1} = (1 - w_q)\overline{q}_k + w_q q_{k+1} \tag{3.1}$$

in which q_{k+1} is the current instantaneous queue length and w_q is the weight factor, or an averaging constant of the low-pass filter. If the queue is empty when a packet arrives, the algorithm takes into account the period when the queue is empty by estimating the number of packets m that could have been transmitted by the router during the idle period of length t_{idle}. After the idle period, the router computes the average queue size as if m packets had arrived to an empty queue,

$$m = f(t_{idle}) \tag{3.2}$$

$$\overline{q}_{k+1} = (1 - w_q)^m \overline{q}_k \tag{3.3}$$

in which $f(t)$ is a linear function of time t.

3.1.2.2 Packet dropping mechanism

The packet dropping mechanism is based on the average queue length $\overline{q}_k$ and the number of packets that already have been accepted since the last dropped packet *cnt*. The average queue length is compared to two thresholds, minimum and maximum thresholds denoted by min_{th} and max_{th}. When the average queue length is less than the minimum threshold, no packets are dropped. When the maximum threshold is

exceeded, every arriving packet is dropped. When the average queue length is between the minimum and maximum thresholds, each arriving packet is dropped with the probability p_b for linear dropping algorithm (Geometric dropping model) or p_{cnt} for non-linear dropping algorithm (Uniform dropping model).

- RED linear dropping algorithm (Geometric dropping model)

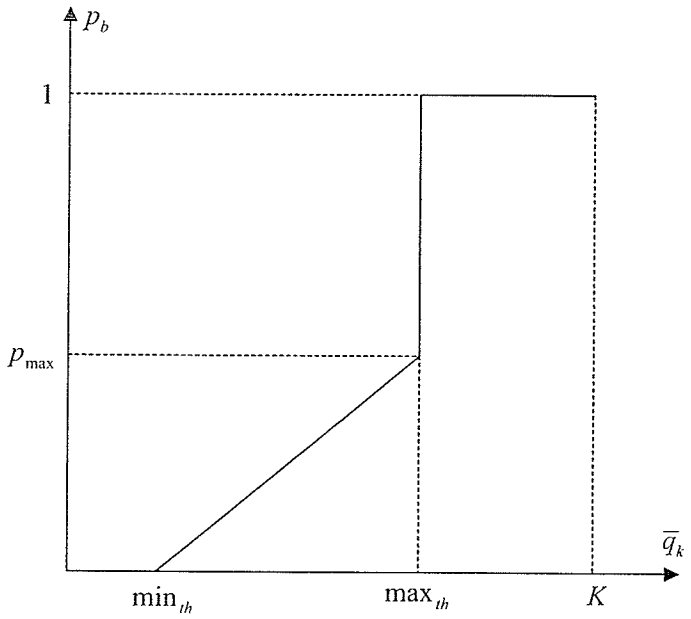

Figure 3.1: Linear dropping algorithm of RED

$$p_b = \begin{cases} 0 & \text{if } \overline{q}_k < min_{th} \\ \frac{(\overline{q}_k - min_{th})p_{max}}{max_{th} - min_{th}} & \text{if } min_{th} \leq \overline{q}_k < max_{th} \\ 1 & \text{if } max_{th} \leq \overline{q}_k \end{cases} \tag{3.4}$$

p_b is a linear function of the average queue length $\overline{q}_k$. Figure 3.1 shows that as

$\overline{q}_k$ varies from min_{th} to max_{th}, the packet dropping probability increases linearly from 0 to a maximum value p_{max}. When the average queue size is greater than the maximum threshold, every arriving packet is dropped. Assuming that the average queue size is constant, let *cnt* be the number of arriving packets between inter-drop packets, because each packet is dropped with probability p_b, we have:

$$Pr\{cnt = n\} = (1 - p_b)^{n-1} p_b \tag{3.5}$$

Thus with linear dropping algorithm, *cnt* has a geometric distribution with parameter p_b, and $E[cnt] = 1/p_b$. We call this model the geometric dropping model.

- RED nonlinear dropping algorithm (Uniform dropping model)

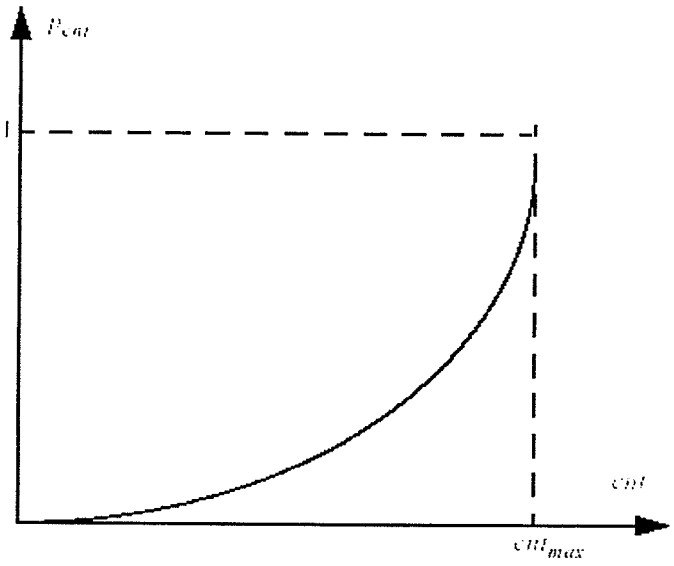

Figure 3.2: Non-linear dropping algorithm of RED

$$p_{cnt} = \frac{p_b}{1 - cnt \times p_b} = \frac{1}{\frac{1}{p_b} - cnt} = \frac{1}{C(\overline{q}_k) - cnt} \tag{3.6}$$

$$C = C(\overline{q}_k) = \frac{1}{p_b} = \frac{max_{th} - min_{th}}{p_{max}(\overline{q}_k - min_{th})} \tag{3.7}$$

where p_{cnt} is a non-linear function of the number of cnt arriving packets between inter-drop packets. Figure 3.2 shows that as cnt varies from 0 to $1/p_b - 1$, the packet dropping probability increases non-linearly from 0 to a maximum value 1. When the number of arriving packets between inter-drop packets is greater than $cnt_{max}(cnt_{max} = 1/p_b - 1)$, every arriving packet is dropped. Assuming that the average queue size is constant, because each packet is dropped with probability p_{cnt}, neglecting the integer constrains, we have

$$Pr[cnt = n] = \begin{cases} \frac{1}{C-(n-1)} \prod_{i=0}^{n-2} (1 - \frac{1}{C-i}) = \frac{1}{C} & 1 \leq n \leq C \\ 0 & C < n \end{cases} \tag{3.8}$$

Thus with non-linear dropping algorithm, cnt has a uniform distribution and $cnt = \{1, 2, ..., C\}$. We call this model the uniform dropping model. Let $\overline{cnt}$ be the expected number of accepted packets between consecutive dropped packets and we have

$$\overline{cnt} = E(cnt) = \sum_{cnt=1}^{\infty} cnt Pr[cnt = n] = \frac{C+1}{2} \approx \frac{C}{2} \tag{3.9}$$

3.1.2.3 Discussion of Two RED Dropping Models

[FJ93] shows that the linear dropping algorithm results in having too many dropped packets close together and having too long an interval between dropped packets. Both of these events can cause global synchronization, with several connections reducing their windows at the same time. [FJ93] also shows that the primary reason for having the counter *cnt* is then to make the flow of congestion indications back to the TCP sources in the steady state as even as possible to reduce the likelihood of having a burst dropped packets resulting in an increased possibility for synchronization among the TCP sources.

3.2 The Modified RED Algorithm

In this section, we give the problem description and then detail our modified RED non-linear algorithm.

3.2.1 Problem Statement

We note that neither RED linear algorithm nor RED non-linear dropping algorithm impose any mandatory separation between successive packet losses. So back to back losses are still possible to cause synchronization of TCP sources. Suppose a random drop queue drops a packet from TCP flow i at time instant t. If there is no mandatory separation between two consecutive packet losses, packets from other TCP flows may also encounter packet drops soon after t. Since TCP reduces its congestion window

in response to a packet drop, such drops can lead to a reduction of the window sizes of multiple TCP connections at around the same time. Imposing a mandatory separation between inter-drop packets can ensure that multiple TCP flows do not reduce their windows simultaneously. Synchronization is harmful since it may lead to under-utilization of the bottleneck bandwidth. May et al. have analyzed the linear dropping algorithm and concluded that the probability of consecutive drops is high for RED linear algorithm, but Floyd et al. [FJ93] suggest to use non-linear algorithm because of its better performance. We propose a modified non-linear algorithm in order to alleviate the possibility of synchronization of TCP flows caused by consecutive packet drops.

3.2.2 Modified Nonlinear RED algorithm

The equation 3.9 shows that the expectation of the number of packets between inter-drop packets is $C/2$. To give a mandatory separation between inter-drop packets, we let $C/2$ as a base separate point. We introduce a latency factor $d(d \geq 1)$ to delay dropping packets until $dC/2$ packets have been accepted by RED gateway and it makes a mandatory separation between inter-drop packets. Also we introduce a compensation factor $m(m \geq 1)$ to modify equation (3.6) in order to compensate the reduced *cnt* range due to having delay factor d. The modified non-linear dropping algorithm is as follows:

$$p_{cnt} = \begin{cases} 0 & \text{if } 0 < cnt \leq \frac{dC}{2}, \quad p_b > 0 \\ p_{cnt} = \frac{\frac{p_b}{m}}{1-\frac{p_b}{m}cnt} = \frac{1}{mC-cnt} & \text{if } \frac{dC}{2} < cnt < cnt_{max}, \quad p_b > 0 \\ 1 & \text{if } cnt_{max} \leq cnt \quad p_b > 0 \end{cases} \tag{3.10}$$

In the equation (3.10), it is possible that p_{cnt} can become negative or greater than 1. When $p_b \neq 0$ and $mC - 1 < cnt < mC$ or $mC < cnt < \infty$, p_{cnt} exceeds 1 or less than zero and $cnt = mC$ results in a division by zero during calculating p_{cnt}. Thus, neglecting the integer constraints, we can derive an upper boundary $cnt_{max}(\overline{q}_k)$ for the average queue size $min_{th} \leq \overline{q}_k < max_{th}$:

$$cnt_{max} = mC - 1 \quad m \geq 1 \tag{3.11}$$

3.2.2.1 Verification of Geometric Dropping Model

In this section, we verify that our modified RED non-linear algorithm still is uniform dropping model because [FJ93] shows that uniform dropping model has better performance in alleviating the synchronization of TCP flows. From our modified algorithm (3.10), when the average queue size is constant, we can get

$$Pr[cnt = n] = \begin{cases} \frac{1}{mC-n} \prod_{i=Cd/2+1}^{n-1} (1 - \frac{1}{mC-i}) = \frac{1}{mC - Cd/2} & Cd/2 + 1 \leq n \leq mC \\ 0 & mC < n \end{cases} \tag{3.12}$$

$$\overline{cnt} = E(cnt) = \sum_{cnt=dC/2+1}^{\infty} cntPr[cnt = n] = \frac{mC + \frac{dC}{2} + 1}{2} \tag{3.13}$$

Equation (3.12) shows that $cnt = \{Cd/2, Cd/2 + 1, ..., mC\}$ still has a uniform distribution, so our modified non-linear dropping algorithm does not change the property of uniform distribution of cnt for RED classical non-linear model. Figure 3.3 shows the cumulative distribution functions (CDFs) for modified nonlinear and classical nonlinear dropping models.

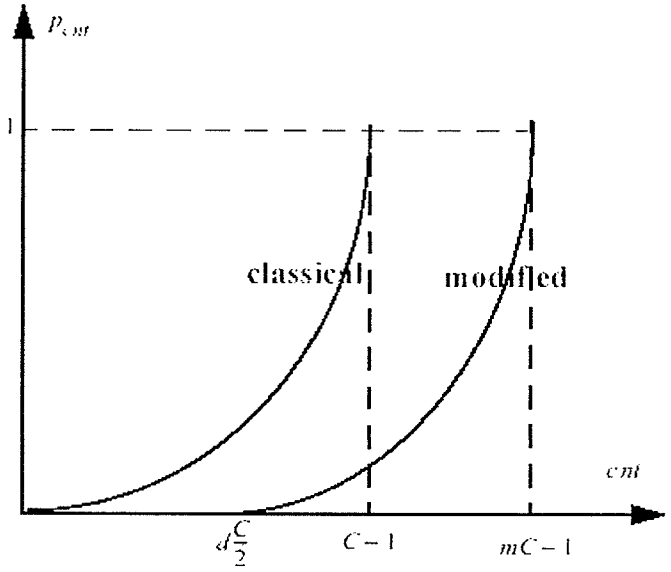

Figure 3.3: CDFs for modified and classical nonlinear RED dropping algorithms

4 Performance Analysis of Different RED Dropping Models

In this chapter, we analyze and compare linear RED, non-linear RED and modified non-linear RED algorithms as well as tail drop in synchronization of TCP flows. We give our mathematical analysis of linear RED, non-linear RED and modified non-linear RED algorithms. We derive a series of closed-form equations which are stationary distribution of instantaneous queue, stationary distribution of consecutive drops, stationary dropping probabilities, expectation and variance of the number of consecutive packet drops and the correlation between stationary average queue occupancy and the offered load based on our approximations. Finally, we give a closed-form solution to set the latency and compensation factors for our modified non-linear RED algorithm.

4.1 Analysis of Linear RED Algorithm and Tail Drop by May et al.

In this section, we give May's analysis models for tail drop and RED linear dropping algorithm in synchronization of TCP flows.

4.1.1 RED router model establishment

The RED router for bursty traffic is modeled as a batch Poisson arrival, in which packets arrive in groups of size B with rate λ shown in Figure 4.1 and for the smooth traffic is modeled as a non-batch Poisson stream shown in Figure 4.2. The processing times of the packets in the router are assumed to be exponential distribution with mean $\mu(-1)$. The offered load for bursty traffic is defined as $\rho = B\lambda/\mu$ and for smooth traffic as $\rho = \lambda/\mu$.

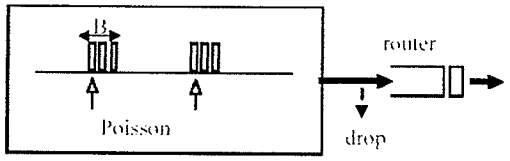

Figure 4.1: Model of RED router with bursty input traffic

4.1.2 Tail Drop

Assume that a drop occurs at time $t = 0$ in a Tail Drop router. Due to the memory-less property of exponential distribution, the next incoming packet is dropped if and only if its arrival time is smaller than the service time of a packet. Thus when a packet is

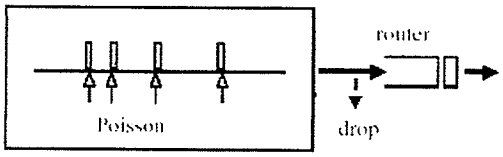

Figure 4.2: Model of RED router with smooth input traffic

dropped, the next packet is dropped with probability p, where

$$p = \int_0^\infty \mu e^{-\mu x}(1 - e^{-\lambda x})dx = \frac{\lambda}{\lambda + \mu} = \frac{\rho}{\rho + 1} \tag{4.1}$$

The number of consecutive drops N_{TD} satisfies

$$p = \{N_{TD} > n\} = p^n \quad \forall n \geq 0 \tag{4.2}$$

Hence

$$E(N_{TD}) = \rho + 1 \tag{4.3}$$

$$Var(N_{TD}) = \rho(\rho + 1) \tag{4.4}$$

4.1.3 RED with instantaneous queue size

Approximation: Consecutive dropped packets are dropped with the same probability.

$\pi(.|\, drop)$ is the stationary distribution of the number of packets in the queue, conditionally to the fact that a drop occurred, the number of consecutive drops N_{RED} satisfies

$$p = \{N_{TD} > n\} = \sum_{K=0}^{K-1} \pi(k|\, drop)d(k)^n \tag{4.5}$$

Using Bay's formula, we have

$$\pi(k|\, drop) = \frac{\pi(k)d(k)}{P(drop)} \quad \forall n \geq 0 \tag{4.6}$$

Then we can get

$$p = \{N_{TD} > n\} = \frac{\sum_{K=0}^{K-1} \pi(k|\, drop)d(k)^{n+1}}{\sum_{K=0}^{K-1} \pi(k)d(k)} \tag{4.7}$$

Furthermore, the mean and variance of N_{RED}

$$E(N_{RED}) = 1 + \frac{\sum_{K=0}^{K-1} \pi(k)\frac{d(k)^2}{1-d(k)}}{\sum_{K=0}^{K-1} \pi(k)d(k)} \tag{4.8}$$

$$Var(N_{RED}) = \frac{\sum_{K=0}^{K-1} \pi(k)(\frac{d(k)}{1-d(k)})^2}{\sum_{K=0}^{K-1} \pi(k)d(k)} \tag{4.9}$$

4.1.4 RED with average queue size

Since the addition of average queue size complicates the modeling and analysis of RED algorithm, May et al. do not give mathematical analysis and just use simulation as the resort.

4.2 Analysis of Non-linear and Modified Non-linear RED Dropping Algorithm

To compare with May's results, the same RED router model is used as above.

4.2.1 Approximations

The RED non-linear algorithms are described by the random process $\{q_k, \overline{q}_k, cnt\}$ and it is computationally very intensive to obtain the stationary distribution even for Poisson arrivals and exponential services times. Therefore, some approximations should be used for models in order to be analytically tractable. Note that the average queue size $\overline{q}_k$ moves slowly when w_q is small (as suggested for practical systems [FJ93]) compared with q_k and cnt, so we assume that under steady state the incoming packets would see the same value for $\overline{q}_k$. Let $\overline{q}_s$ denote the stationary average queue occupancy and then $C = C(\overline{q}_s) = \frac{1}{p_b(\overline{q}_s)}$ is a constant when the system is under steady state. We assume that $\overline{q}_s$ equals to the average stationary queue occupancy when w_q is small. It is computationally much easier to obtain the stationary distribution of the process $\{q_k, cnt\}$ than $\{q_k, \overline{q}_k, cnt\}$ because of the reduction in the dimensionality.

4.2.2 The derivation of the stationary distribution of $\{q_k, cnt\}$

The number of packets buffered in the queue and the number of accepted packets between inter-drop packets define a 2-dimension Markov chain $\{q_k, cnt\}$. Since the arrivals are Poisson process, according to PASTA property, the stationary distribution

of $\{q_k, cnt\}$ is the same as that of the corresponding continuous time system with the state transition diagram and transition rates as shown in Figure 4.3. But it is hard to obtain the closed-form solution of the stationary distribution $\pi_{q_k,cnt}$ of the process $\{q_k, cnt\}$.

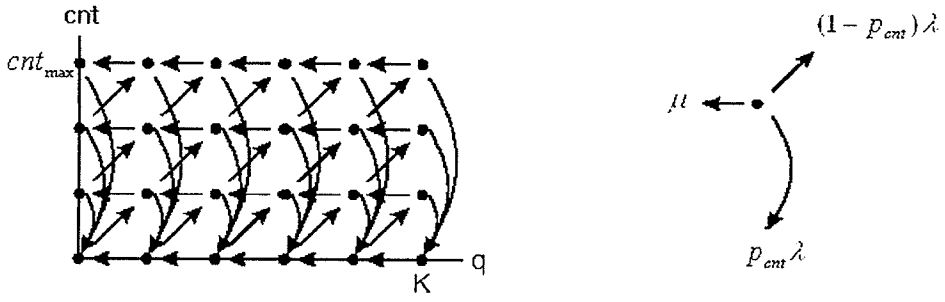

Figure 4.3: The state transition diagram and transition rates for the process $\{q_k, cnt\}$

We note that the process $\{q_k\}$ and $\{cnt\}$ constitute a Markov chain individually and it is easier to compute their stationary distributions by a closed-form solution. So we will decompose the 2-dimension Markov chain into two 1-dimension Markov chains $\{q_k\}$ and $\{cnt\}$.

4.2.2.1 The stationary distribution of $\{cnt\}$

The number of accepted packets between inter-drop packets defines a Markov chain $\{cnt\}$ and its state transition diagram is shown in Figure 4.4. Let π_{cnt} denote the stationary distribution of $\{cnt\}$.

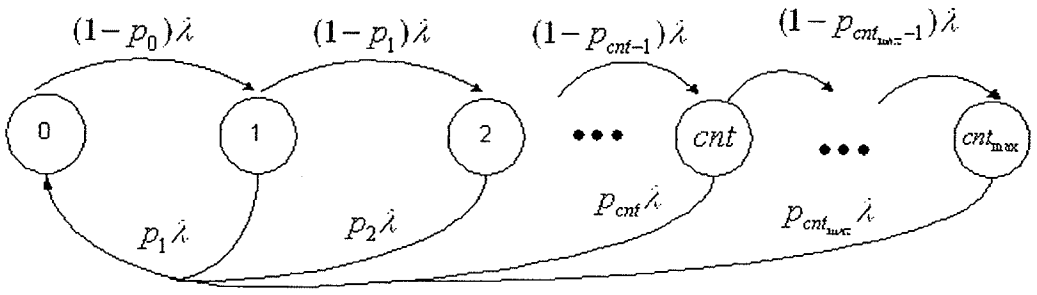

Figure 4.4: State transition diagram for the number of accepted packets between inter-drop packets

Equating flow out to flow in gives:

$$\begin{aligned}
(1-p_0)\pi_0\lambda &= p_1\pi_1\lambda + p_2\pi_2\lambda + \cdots + p_{cnt_{max}}\pi_{cnt_{max}}\lambda \\
(1-p_0)\pi_0\lambda &= p_1\pi_1\lambda + (1-p_1)\pi_1\lambda = \pi_1\lambda \\
(1-p_1)\pi_1\lambda &= p_2\pi_2\lambda + (1-p_2)\pi_2\lambda = \pi_2\lambda \\
&\vdots \\
(1-p_{cnt-1})\pi_{cnt-1}\lambda &= \pi_{cnt}\lambda
\end{aligned} \tag{4.10}$$

Rewriting, we get

$$\begin{aligned}
(1-p_0)\pi_0 &= p_1\pi_1 + p_2\pi_2 + \cdots + p_{cnt_{max}}\pi_{cnt_{max}} \\
\pi_1 &= (1-p_0)\pi_0 \\
\pi_2 &= (1-p_0)(1-p_1)\pi_0 \\
&\vdots \\
\pi_{cnt} &= \pi_0 \prod_{n=0}^{cnt-1}(1-p_n) \ (cnt \geq 1)
\end{aligned} \tag{4.11}$$

Since probabilities must sum to 1, i.e. $\sum_{cnt=0}^{cnt_{max}} = 1$, we have:

$$\pi_{cnt} = \frac{\prod_{n=0}^{cnt-1}(1-p_n)}{1+\sum_{n=1}^{cnt_{max}}\prod_{cnt=0}^{n-1}(1-p_{cnt})}$$

$$\pi_0 = \frac{1}{1+\sum_{n=1}^{cnt_{max}}\prod_{cnt=0}^{n-1}(1-p_{cnt})} \tag{4.12}$$

From the classical and our modified RED algorithms (3.10) and equation (3.6), (4.11) and (4.12), we can derive equation (4.12) further to get their definite expressions.

For the classical non-linear RED algorithm, we have

$$\begin{aligned}
\sum_{cnt=0}^{cnt_{max}} \pi_{cnt} &= \pi_0 + \pi_1 + \cdots + \pi_{cnt} \\
&= \pi_0(1 + (1-p_0) + (1-p_0)(1-p_1) + \cdots + \\
&(1-p_0)(1-p_1)\cdots(1-p_{cnt_{max}-1})) \\
&= \pi_0(1 + \frac{1}{C}(C-1+C-2+\cdots+C-cnt_{max})) \\
&= \pi_0(1 + \frac{1}{C}\frac{cnt_{max}(C-1+C-cnt_{max})}{2}) \\
&= \pi_0\frac{C+1}{2} = 1
\end{aligned} \tag{4.13}$$

$$\Rightarrow \pi_0 = \frac{2}{C+1} \tag{4.14}$$

$$\pi_{cnt} = \pi_0 \prod_{n=0}^{cnt-1}(1-p_n) = \frac{2(C-cnt)}{C(C+1)} \qquad (0 \le cnt \le cnt_{max}) \tag{4.15}$$

For our modified RED algorithm, we have

$$\begin{aligned}
&\sum_{cnt=0}^{cnt_{max}} \pi_{cnt} = \pi_0 + \pi_1 + \cdots + \pi_{cnt} \\
&= \pi_0(1 + (1-p_0) + (1-p_0)(1-p_1) + \cdots + \\
&(1-p_0)(1-p_1)\cdots(1-p_{cnt_{max}-1})) \\
&= \pi_0(1 + \frac{dC}{2} + 1 + (1-p_{\frac{dC}{2}+1}) + (1-p_{\frac{dC}{2}+1}) \\
&(1-p_{\frac{dC}{2}+2}) + \cdots + (1-p_{\frac{dC}{2}+1})(1-p_{\frac{dC}{2}+2}) \quad \cdots (1-p_{cnt_{max}-1})) \\
&= \pi_0(1 + (\frac{dC}{2}+1) + \frac{1}{mC - \frac{dC}{2} - 1}(mC - \frac{dC}{2} - 1 + mC - \frac{dC}{2} - 2 \\
&+ \cdots + mC - cnt_{max}) \\
&= \pi_0(1 + \frac{dC}{2} + \\
&\frac{1}{mC - \frac{dC}{2} - 1}\frac{(cnt_{max} - \frac{dC}{2})(mC - \frac{dC}{2} - 1 + mC - \frac{dC}{2} - (cnt_{max} - \frac{dC}{2}))}{2} \\
&= \pi_0(1 + \frac{dC}{2} + \frac{(mC - \frac{dC}{2})(mC - \frac{dC}{2} - 1)}{2(mC - \frac{dC}{2} - 1)} \\
&= \frac{\pi_0(mC + \frac{dC}{2} + 2)}{2} = 1
\end{aligned} \tag{4.16}$$

$$\Rightarrow \pi_0 = \frac{2}{mC + \frac{dC}{2} + 2} \tag{4.17}$$

$$\pi_{cnt} = \pi_0 \prod_{n=\frac{dC}{2}+1}^{cnt-1} (1-p_n) = \frac{2(mC - cnt)}{(mC + \frac{dC}{2} + 2)(mC - \frac{dC}{2} - 1)} \qquad (\frac{dC}{2} + 1 \leq cnt \leq cnt_{max}) \tag{4.18}$$

4.2.2.2 The stationary drop probability

Let p_d denote the stationary drop probability of a packet in RED gateway and by using clasical and modified RED algorithm (3.6) and (3.10), and equation (4.15) and (4.18), we can approximate p_d as follows:

$$p_d = \pi_{cnt} p_{cnt} = \begin{cases} \frac{2}{\overline{C}(\overline{q}_s)(\overline{C}(\overline{q}_s)+1)} & \textit{nonlinear RED model} \\ \\ \frac{2}{(m\overline{C}(\overline{q}_s)+\frac{d\overline{C}(\overline{q}_s)}{2}+2)(m\overline{C}(\overline{q}_s)-\frac{d\overline{C}(\overline{q}_s)}{2}-1)} & \textit{modified nonlinear RED model} \end{cases} \tag{4.19}$$

We conclude from equation (4.19) that the approximated stationary drop probability of a packet in RED gateway is a constant, i.e. it is independent of the number of accepted packets between inter-drop packets and the stationary drop probability are equal at the point where the drop actually occurs in RED gateway.

4.2.2.3 The stationary distribution of the process $\{q_k\}$

Let q denote the queue length, i.e. the number of packets in the RED gateway (including the one in service) and let K denote the maximum queue size and let π_q denote the stationary distribution of $\{q_k\}$. Then the process $\{q_k\}$ defines a birth-death Markov chain shown in Figure 4.5.

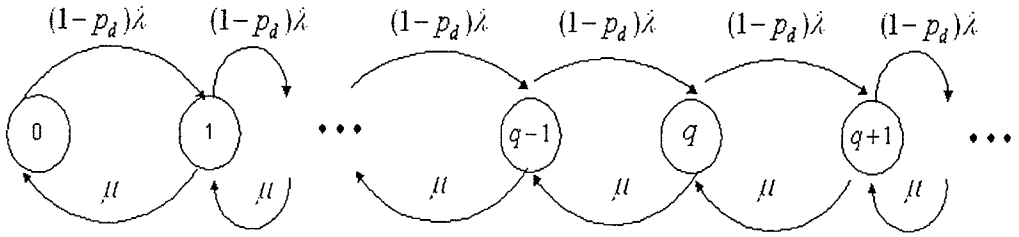

Figure 4.5: The state transition diagram of $\{q_k\}$

To begin, consider a state q in the queue. From state q, system goes to state $q-1$ if a service is completed or to $q+1$ if an arrival occurs without packet drop. By equating the rate at which the process leaves a state with the rate at which it enters that state, we can obtain:

$$\begin{aligned}
&(1-p_d)\lambda\pi_0 = \pi_1\mu \\
&((1-p_d)\lambda+\mu)\pi_1 = (1-p_d)\lambda\pi_0 + \pi_2\mu \\
&\quad\vdots \\
&((1-p_d)\lambda+\mu)\pi_q = (1-p_d)\lambda\pi_{q-1} + \pi_{q+1}\mu
\end{aligned} \tag{4.20}$$

By adding to each equation the equation preceding it, we have

$$\begin{aligned}
(1-p_d)\lambda\pi_0 &= \pi_1\mu \\
(1-p_d)\lambda\pi_1 &= \pi_2\mu \\
&\vdots \\
(1-p_d)\lambda\pi_q &= \pi_{q+1}\mu
\end{aligned} \tag{4.21}$$

Solving in terms of π_0 yields

$$\begin{aligned}
\pi_1 &= (1-p_d)\tfrac{\lambda}{\mu}\pi_0 = \rho(1-p_d)\pi_0 \\
\pi_2 &= (1-p_d)\tfrac{\lambda}{\mu}\pi_1 = \rho^2(1-p_d)^2\pi_0 \\
&\vdots \\
\pi_q &= (1-p_d)\tfrac{\lambda}{\mu}\pi_{q-1} = \rho^q(1-p_d)^q\pi_0
\end{aligned} \tag{4.22}$$

And by using the fact that $\sum_{q=0}^{K}\pi_q = 1$, we can obtain

$$\begin{aligned}
\pi_0 &= \tfrac{1-\rho(1-p_d(\overline{q}_s))}{1-\rho^{k+1}(1-p_d(\overline{q}_s))^{K+1}} \\
&\qquad\qquad (0 \le q \le K) \\
\pi_q &= \tfrac{\rho^q(1-p_d(\overline{q}_s))^q(1-\rho(1-p_d(\overline{q}_s)))}{1-\rho^{K+1}(1-p_d(\overline{q}_s))^{K+1}}
\end{aligned} \tag{4.23}$$

4.2.3 The analysis of consecutive packet drops

4.2.3.1 The derivation of the expectation and variance of the number of consecutive drops

As in above, we get the stationary distributions of $\{cnt\}$ and $\{q_k\}$ and then we derive the stationary drop probabilities of classical nonlinear RED and modified nonlinear RED algorithms. From the stationary drop probability, we can obtain the probability of the number of consecutive drops in a RED gateway. Let N_{drop} denote the number of consecutive drops and we can get

$$p\{N_{drop} = n\} = (p_d(\overline{q}_s))^{n-1}(1 - p_d(\overline{q}_s)) \tag{4.24}$$

From equation (4.24) and (4.19), we have

$$\begin{aligned} p\{N_{drop} > n\} &= \sum_{n=n+1}^{\infty} (p_d(\overline{q}_s))^{n-1}(1 - p_d(\overline{q}_s)) \\ &= (P_d(\overline{q}_s))^n \\ &= \begin{cases} (\frac{2}{C(\overline{q}_s)(C(\overline{q}_s)+1)})^n & \textit{nonlinear RED model} \\ (\frac{2}{(mC(\overline{q}_s)+\frac{dC(\overline{q}_s)}{2}+2)(mC-\frac{dC(\overline{q}_s)}{2}-1)})^n & \textit{modified nonlinear RED model} \end{cases} \end{aligned} \tag{4.25}$$

From equation (4.24) and (4.19) , we can derive the expectation and the variance of the number of consecutive drops:

$$E(N_{drop}) = \frac{1}{1 - p_d(\overline{q}_s)}$$

$$= \begin{cases} \frac{1}{1 - \frac{2}{C(\overline{q}_s)(C(\overline{q}_s)+1)}} & \textit{nonlinear RED model} \\ \\ \frac{1}{1 - \frac{2}{(mC(\overline{q}_s) + \frac{dC(\overline{q}_s)}{2} + 2)(mC(\overline{q}_s) - \frac{dC(\overline{q}_s)}{2} - 1)}} & \textit{modified nonlinear RED model} \end{cases} \tag{4.26}$$

$$Var(N_{drop}) = \frac{p_d(\overline{q}_s)}{(1 - p_d(\overline{q}_s))^2}$$

$$= \begin{cases} \frac{\frac{2}{C(\overline{q}_s)(C(\overline{q}_s)+1)}}{(1 - \frac{2}{C(\overline{q}_s)(C(\overline{q}_s)+1)})^2} & \textit{nonlinear RED model} \\ \\ \frac{\frac{2}{(mC(\overline{q}_s) + \frac{dC(\overline{q}_s)}{2} + 2)(mC(\overline{q}_s) - \frac{dC(\overline{q}_s)}{2} - 1)}}{(1 - \frac{2}{((mC(\overline{q}_s) + \frac{dC(\overline{q}_s)}{2} + 2)(mC(\overline{q}_s) - \frac{dC(\overline{q}_s)}{2} - 1)})^2} & \textit{modified nonlinear RED model} \end{cases} \tag{4.27}$$

4.2.3.2 The correlation between stationary average queue occupancy $\overline{q}_s$ and the offered load ρ

The average queue occupancy is the function of the offered load ρ, so the distribution of the number of consecutive drops also is the function of the offered load. Let $E_\pi(\pi_q)$ denote the expectation of the stationary queue occupancy. According to our assumption that the stationary average queue occupancy equals average stationary queue occupancy when w_q is small in section 5.2.1, we can get

$$E_\pi(\pi_q(\overline{q}_s)) = \overline{q}_s \tag{4.28}$$

From equation (4.23), we can get the generating function of $\{\pi_q\}$

$$\begin{aligned} g(z) &= \sum_{q=0}^{\infty} \pi_q z^q \\ &= \frac{1-\rho(1-p_d(\overline{q}_s))}{1-\rho^{K+1}(1-p_d(\overline{q}_s))^{K+1}} \sum_{q=0}^{\infty} \rho^q (1-p_d(\overline{q}_s))^q z^q \\ &= \frac{1-\rho(1-p_d(\overline{q}_s))}{1-\rho^{K+1}(1-p_d(\overline{q}_s))^{K+1}} \frac{1-(\rho(1-p_d(\overline{q}_s))z)^{K+1}}{1-\rho(1-p_d(\overline{q}_s))z} \end{aligned} \tag{4.29}$$

From equation (4.29), we can get the expectation of $\{\pi_q\}$

$$\begin{aligned} E_\pi(\pi_q(\overline{q}_s)) &= \frac{dg(z)}{dz}\,|_{z=1} \\ &= \frac{\rho(1-p_d(\overline{q}_s))}{1-\rho(1-p_d(\overline{q}_s))} - (K+1)\frac{(\rho(1-p_d(\overline{q}_s)))^{K+1}}{1-(\rho(1-p_d(\overline{q}_s)))^{K+1}} \end{aligned} \tag{4.30}$$

From equations (4.30), (3.7) and (4.19), $\overline{q}_s$ can be solved numerically, given the offered load ρ.

4.2.4 Setting latency factor d and compensation factor m

Note that stationary drop probability p_d less than 1. From equation (4.19), we have

$$\begin{aligned} &(mC + \tfrac{dC}{2} + 2)(mC - \tfrac{dC}{2} - 1) > 2 \\ &(mC - \tfrac{dC}{2} - 1) > 0 \end{aligned} \tag{4.31}$$

Solving the inequality (4.31), we get

$$m > \sqrt{\frac{d^2}{2} + \frac{d}{2C} + \frac{17}{4C^2}} - \frac{1}{2C} \tag{4.32}$$

From equation (3.7) $C = 1/p_b$, we can derive the minimum value of $C_{min} = 1$. Hence, the minimum value of m is as follows:

$$m_{min} = \sqrt{\frac{d^2}{2} + \frac{d}{2} + \frac{17}{4}} - \frac{1}{2} \tag{4.33}$$

We note that $cnt_{max} \leq max_{th}$, from equation (3.11), we have

$$cnt_{max} = mC - 1 \leq max_{th} \tag{4.34}$$

$$\Rightarrow m_{max} = \frac{max_{th} + 1}{C} \tag{4.35}$$

We will choose the latency factor d and the compensation factor m from formula (4.33) and (4.35) in numerical analysis.

5 Performance Evaluation of Different RED Dropping Models

In this chapter, we evaluate and compare the performance of different RED dropping models using the analysis presented in chapter 4. The various numerical scenarios are investigated for comparing the performance of linear RED, non-linear RED and the modified non-linear RED algorithms in the characteristics of consecutive packet drops.

5.1 Analysis and Comparison of Distributions of Consecutive Drops

In this section, we compare May's result with non-linear and modified non-linear algorithms presented in the preceding chapters in the distribution of consecutive drops.

5.1.1 Linear RED dropping model by May et al.

In this subsection, we show May's analysis results in global synchronization of TCP flows.

5.1.1.1 RED with instantaneous queue size

May et al compare and analyze the distributions of consecutive drops with linear RED dropping model and Tail Drop for an offered load of $\rho = 2$ and RED parameters are chosen to be following: $K = 40$, $min_{th} = 20$, $max_{th} = 40$, $p_{max} = 1$ and $\rho = 2$. Figure 5.1 shows the distribution of the number of consecutive drops. Table 5.1 gives the expectations and variances of consecutive drops of Tail Drop and linear RED model. They conclude that the linear RED has lower consecutive drops than Tail Drop when drop probability is counted with instantaneous queue size.

	expectation	variance
Tail Drop	3	6
Linear RED	2.3	4.1

Table 5.1: The expectation and variance of the number of consecutive drops for an offered load of $\rho = 2$ with instantaneous size

5.1.1.2 RED with average queue size

Since the addition of average queue size complicates the modeling and analysis of RED algorithm, May et al. do not give mathematical analysis and just use simulation results.

The result shows that RED significantly increases the mean and variance of the number of consecutive drops, especially when it is close to its recommended value of 0.002

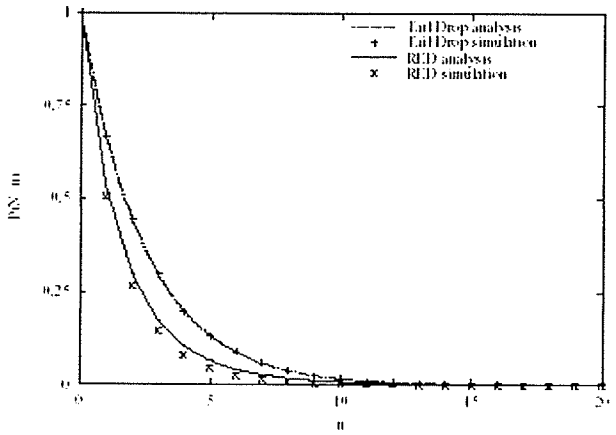

Figure 5.1: Distribution of the number of consecutive drops for an offered load of $\rho = 2$ with instantaneous size

[FJ93]. This suggests that deploying RED may in fact contribute to the synchronization of TCP flows. Note that the conclusion drawn above is based upon RED linear dropping algorithm, but [FJ93] have already shown that the non-linear dropping model (uniform dropping scheme) yields better performance in terms of avoiding TCP synchronization than the linear dropping model. Hence, the result made by May et al. may not be valid. We analyze and compare non-linear and modified non-linear RED algorithm with linear RED algorithm in next section.

5.1.2 The nonlinear and modified nonlinear RED dropping models

In this section, we give the numerical results of nonlinear and modified nonlinear RED dropping models in consecutive drops. To compare with May's results, we use the same

	expectation	variance
Tail Drop	3	6
Linear RED $w = 0.1$	5.9	40
Linear RED $w = 0.01$	7.7	170
Linear RED $w = 0.001$	7.2	190

Table 5.2: The expectation and variance of the number of consecutive drops for an offered load of $\rho = 2$ with average queue size

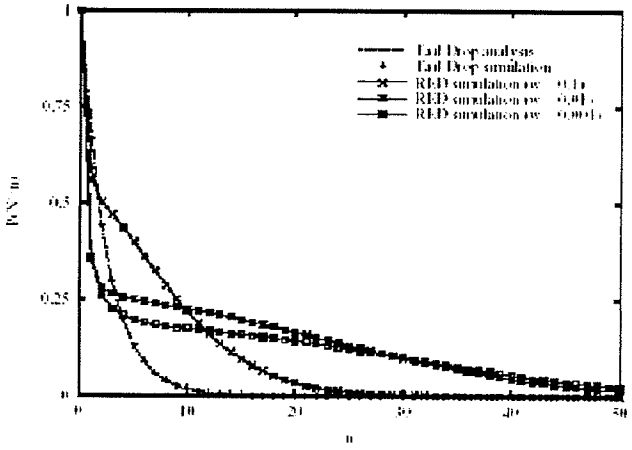

Figure 5.2: Distribution of the number of consecutive drops for an offered load of $\rho = 2$ with average queue size

system parameters as May's. The system parameters are chosen in this section to be following: $K = 40$, $min_{th} = 20$, $max_{th} = 40$, $p_{max} = 1$ and $\rho = 2$.

5.1.2.1 Solving the stationary average queue occupancy $\overline{q}_s$

According to equation (4.30), (3.7) and (4.19), Fig. 5.3, Fig. 5.4, and Fig. 5.5 plot the $\overline{q}_s$ as a function of the load ρ for different models. We can get the stationary average queue occupancy $\overline{q}_s$ by given ρ. Fig. 5.3 gives the value of the $\overline{q}_s$ as a function of the ρ for classical non-linear RED model and Fig. 5.4 and Fig. 5.5 give the value of the $\overline{q}_s$ as a function of the ρ for modified non-linear RED model with different latency factors and compensation factors. From Fig. 5.3, Fig. 5.4, and Fig. 5.5, we can get Table 5.3.

	ρ	$\overline{q}_s$
classical non-linear model	2	32
modified non-linear model($d = 1$, $m = 2$)	2	35
modified non-linear model($d = 2$, $m = 3$)	2	38

Table 5.3: The values of $\overline{q}_s$ for an offered load of $\rho = 2$ for different models

We can compute the distribution of the number of consecutive drops and its expectation and variance by Table 5.3, equation (3.7), (4.19), (4.26) and (4.27).

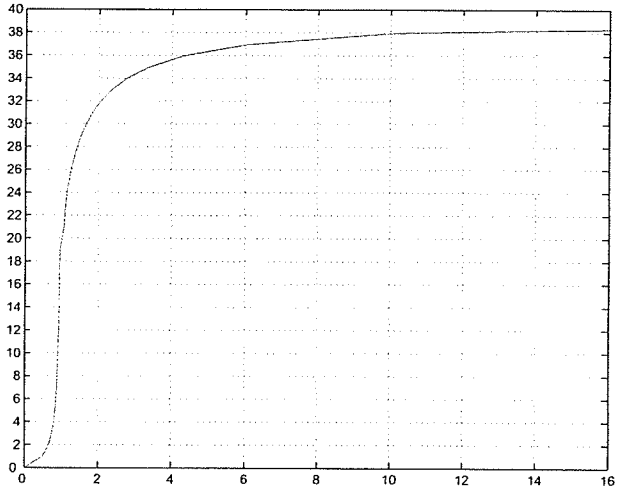

Figure 5.3: Value of the $\overline{q}_s$ as a function of the ρ for classical non-linear RED model($min_{th} = 20$, $max_{th} = 40$)

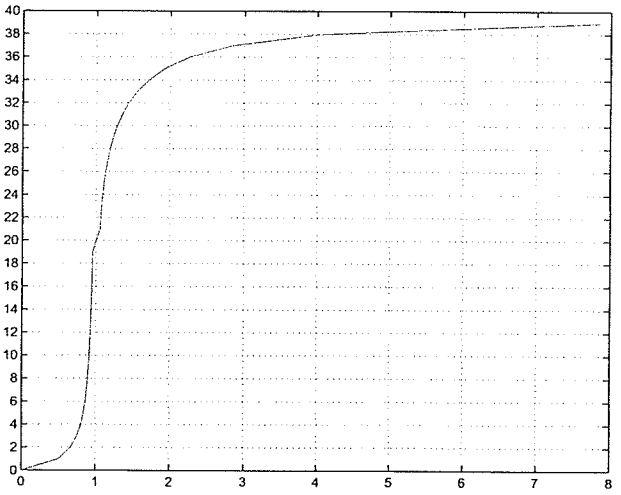

Figure 5.4: Value of the $\overline{q}_s$ as a function of the ρ for modified non-linear RED model($d = 1$, $m = 2$)

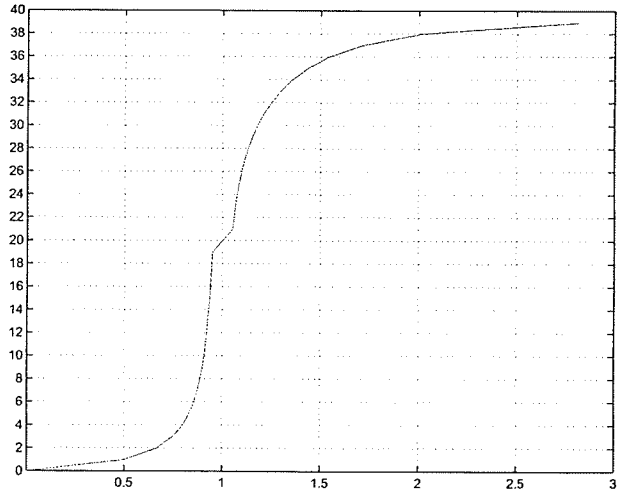

Figure 5.5: Value of the $\overline{q}_s$ as a function of the ρ for modified non-linear RED model($d = 2$, $m = 3$)

5.1.2.2 The distributions of the number of consecutive drops

In this section, we illustrate distributions of the number of consecutive drops in RED gateway for non-linear model and our modified non-linear model. We also shows the expectation and variance of the number of consecutive drops.

	mean	variance
Non-linear model	1.8	1.5
modified non-linear model($d = 1$, $m = 2$)	1.6	0.96
modified non-linear model($d = 2$, $m = 3$)	1.3	0.45

Table 5.4: mean and variance of the number of consecutive drops for an offered load $\rho = 2$ with different nonlinear RED dropping models

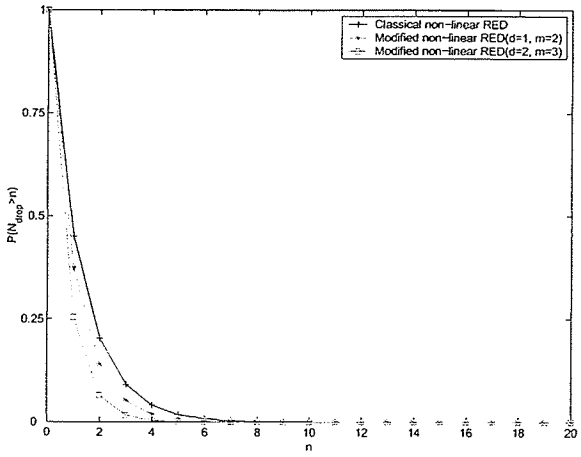

Figure 5.6: Distribution of the number of consecutive drops for an offered load of $\rho = 2$ for classical non-linear RED model and the modified non-linear RED model

5.1.3 Analysis and Comparison of different dropping models

In this subsection, we compare the performance of consecutive drops in RED gateway for different RED dropping models which are classical linear model (May's analysis model), classical non-linear model and our modified non-linear model.

Fig. 5.7 gives the the distribution of the number of consecutive drops for linear RED model from May's analysis, classical non-linear RED model and the modified RED model from our analysis. Table 5.5 gives the mean and variance of the number of consecutive drops obtained by May et al. and the mean and variance of the number of consecutive drops for classical and our modified non-linear RED algorithms.

	mean	variance
Tail Drop	3	6
linear RED $w = 1$	2.3	4.1
Linear RED $w = 0.1$	5.9	40
Linear RED $w = 0.1$	7.7	170
Linear RED $w = 0.1$	7.2	190
classical non-linear model	1.8	1.5
modified non-linear model($d = 1$, $m = 2$)	1.6	0.96
modified non-linear model($d = 2$, $m = 3$)	1.3	0.45

Table 5.5: mean and variance of the number of consecutive drops for an offered load $\rho = 2$ with different models

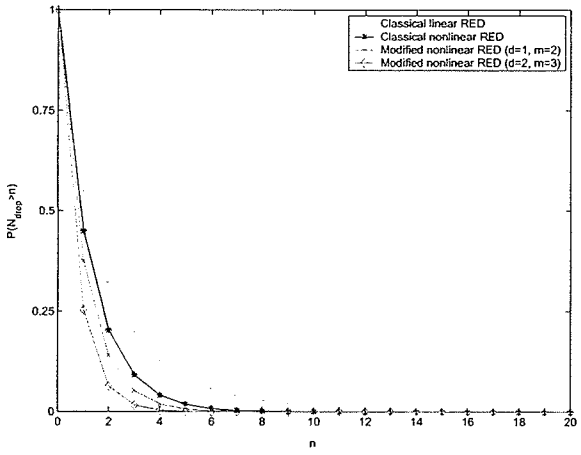

Figure 5.7: Comparison of distributions of the number of consecutive drops of different RED dropping models for an offered load of $\rho = 2$

5.2 Evaluation of Distributions of Consecutive Drops for More Cases

In this section, we give more numerical examples to show the performance of modified RED in consecutive drops.

5.2.1 Different offered loads

In this section, we give another example in which the offered load is different from above and the other parameters are the same. RED parameters are chosen to be following: $K = 40$, $min_{th} = 20$, $max_{th} = 40$, $p_{max} = 1$ and $\rho = 1.5$. 5.6 and Fig. 5.9 give the results.

	mean	variance
Tail Drop	2.5	3.8
linear RED	1.9	2.1
classical non-linear model	1.6	1.05
modified non-linear model($d = 1$, $m = 2$)	1.3	0.48
modified non-linear model($d = 2$, $m = 3$)	1.3	0.23

Table 5.6: mean and variance of the number of consecutive drops for an offered load $\rho = 1.5$ with different models

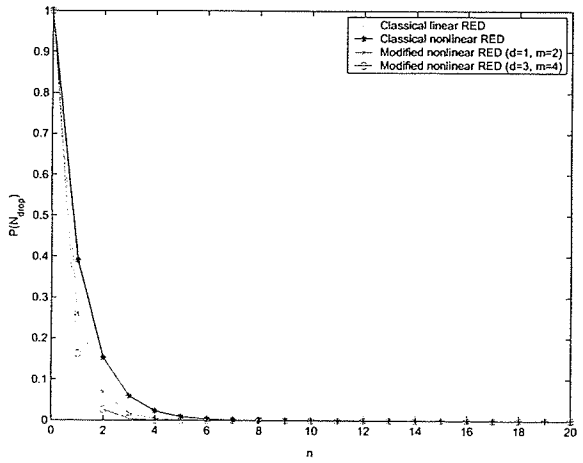

Figure 5.8: Distribution of the number of consecutive drops for an offered load of $\rho = 1.5$ with different RED dropping models

5.2.2 Different latency and compensation factors

In this section, we give different latency and compensation factors from above to show the performance of the modified nonlinear RED algorithm. The system parameters are chosen in this section to be following: $K = 40$, $min_{th} = 20$, $max_{th} = 40$, $p_{max} = 1$ and $\rho = 2$, $d = 3$, $m = 4$. 5.7 and Fig. 5.10 give the results.

5.2.3 Different queue thresholds

In this section, we give different thresholds from above to show the performance of the modified nonlinear RED algorithm. The system parameters are chosen in this section to be following: $K = 60$, $min_{th} = 20$, $max_{th} = 60$, $p_{max} = 1$, $\rho = 2$, $d = 2$ and $m = 3$.

	mean	variance
Tail Drop	3	6
linear RED	2.3	4.1
classical non-linear model	1.8	1.5
modified non-linear model($d = 1$, $m = 2$)	1.6	0.96
modified non-linear model($d = 3$, $m = 4$)	1.2	0.19

Table 5.7: mean and variance of the number of consecutive drops for an offered load $\rho = 2$ with different latency factors

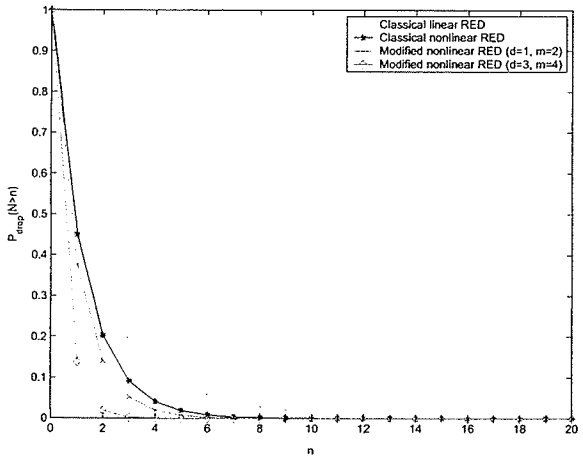

Figure 5.9: Distribution of the number of consecutive drops for an offered load of $\rho = 2$ with different latency factors

5.2.3.1 Solving the stationary average queue occupancy $\overline{q}_s$

According to equation (4.30), (3.7) and (4.19), Fig.5.3, Fig.5.4, and Fig.5.5 plot the $\overline{q}_s$ as a function of the load ρ. We can get the stationary average queue occupancy $\overline{q}_s$ by given ρ. Fig.5.10 gives the value of the $\overline{q}_s$ as a function of the ρ for classical linear RED model and Fig.5.11 gives the value of the $\overline{q}_s$ as a function of the ρ for classical nonlinear RED model and Fig.5.12 gives the value of the $\overline{q}_s$ as a function of the ρ for modified non-linear RED model with $d = 2$ and $m = 3$. From Fig.5.10, Fig.5.11, and Fig.5.12, we can get Table 5.8.

	ρ	$\overline{q}_s$
classical linear model	2	48
Classical non-linear model	2	54
modified non-linear model($d = 2$, $m = 3$)	2	58

Table 5.8: The values of $\overline{q}_s$ for an offered load of $\rho = 2$ with different thresholds

We can compute the distribution of the number of consecutive drops and its expectation and variance by Table 5.8, equation (3.7), (4.19), (4.26) and (4.27).

5.2.4 Analysis and Comparison of Different Dropping Models

The numerical results show that our modified non-linear RED algorithm has much lower consecutive dropping probability than classical linear (May's analysis model)

	mean	variance
Tail Drop	3	6
linear RED	2.3	3.6
classical non-linear model	1.8	1.4
modified non-linear model($d = 2$, $m = 3$)	1.4	0.5

Table 5.9: mean and variance of the number of consecutive for an offered load $\rho = 2$ with different thresholds

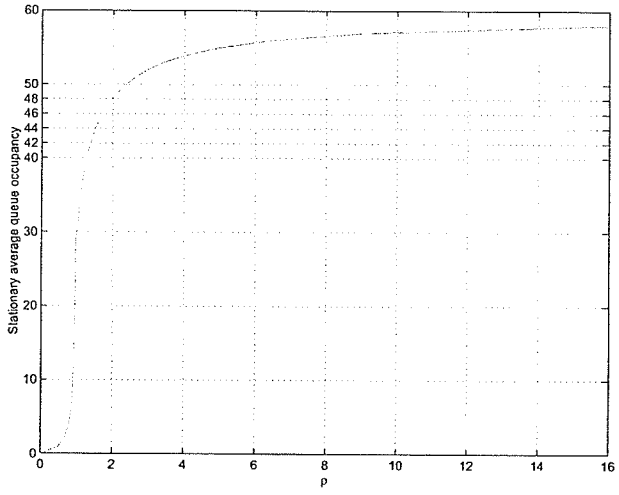

Figure 5.10: Value of the $\overline{q}_s$ as a function of the ρ for classical linear RED model with different thresholds

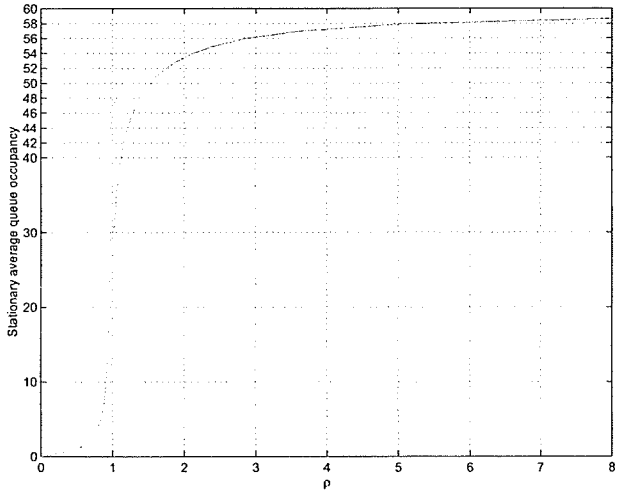

Figure 5.11: Value of the $\overline{q}_s$ as a function of the ρ for classical non-linear RED model with different thresholds

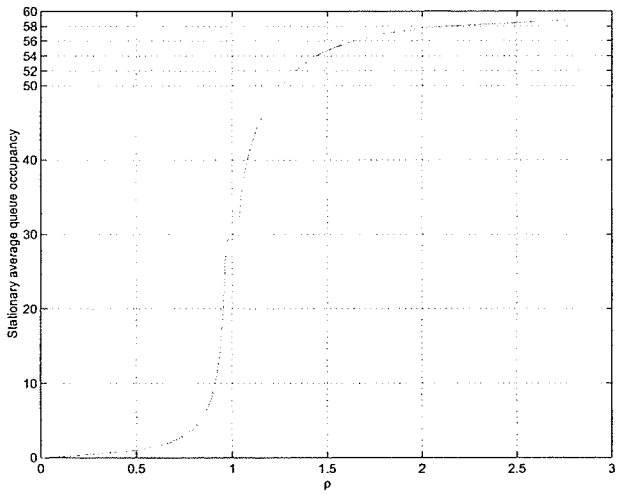

Figure 5.12: Value of the $\overline{q}_s$ as a function of the ρ for modified non-linear RED model with different thresholds

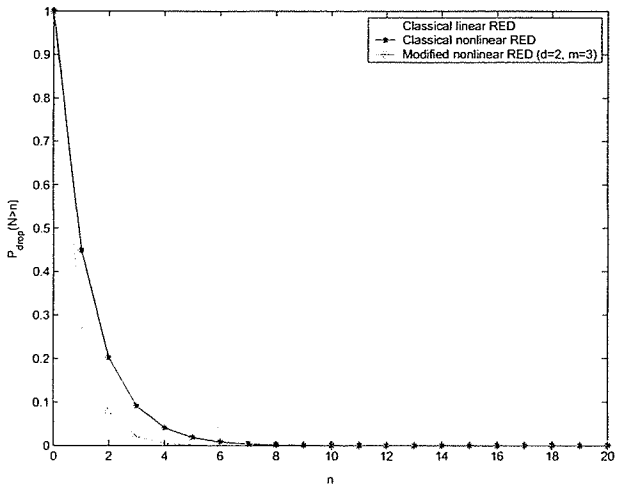

Figure 5.13: Distribution of the number of consecutive drops for an offered load of $\rho = 2$ for different RED dropping models with different thresholds

and non-linear RED algorithms in different cases. Fig. 5.8, Fig. 5.9 and Fig. 5.13 demonstrate that the bigger the latency , the lower the probability of the consecutive drops.

6 Conclusions and Future Work

This chapter gives our conclusions based on numerical results in chapter 5 and then highlights the major contributions achieved by this work as well as the future directions for this research.

In this research, we challenge the claim of [MBB00] in global synchronization of TCP flows in RED gateway and then develop a modified nonlinear RED algorithm to overcome the weakness of high consecutive drops of RED. Furthermore, we evaluate and analyze the behavior and performance of linear RED and nonlinear RED algorithms in global synchronization of TCP flows. We give a mathematical analysis with closed form solution. The numerical results show that our modified non-linear RED algorithm has lowest consecutive dropping probability among the Tail Drop, classical linear, non-linear RED and modified nonlinear RED algorithms. Also, we see that linear RED drop algorithm has the highest consecutive dropping probability compared with other three schemes and it means that RED non-linear drop strategy is better than RED linear drop strategy in avoiding global synchronization of TCP flows. This conclusion coincides with that of [FJ93]. We conclude that our modified RED non-linear algorithm can avoid the synchronization of TCP flows more efficiently.

6.1 Major Contributions

This thesis has focused on solving global synchronization of TCP flows in RED gateway. We challenge May's claim and analyze the nonlinear RED algorithm. Then we develop and analyze a modified nonlinear RED algorithm to overcome the weakness of high consecutive drops of RED and give a mathematical analysis with closed form solution. The major contributions are as follows:

1. Propose a modified non-linear RED algorithm to improve RED performance in global synchronization of TCP flows.

   We introduce latency and compensation factors to modify nonlinear RED algorithm. We note the weakness of RED algorithms in global synchronization of TCP flows. Neither RED linear algorithm nor RED non-linear dropping algorithm impose any mandatory separation between successive packet losses, so back to back losses are still possible to cause synchronization of TCP source. We introduce a latency factor to let RED gateway delay dropping packets until a definite number of packets has been accepted by the RED gateway. The latency factor makes a mandatory separation between consecutive drops to alleviate the probability of consecutive drops in order to avoid the global synchronization of TCP flows efficiently. I also introduce a compensation factor to compensate the reduced range of the number of consecutive drops varying.

2. Establish the analytic model and solve it with closed form solutions.

   We found and derived the mathematical solutions with closed form equations for

nonlinear and modified nonlinear RED models. These solutions are as follows:

- Derivation of the closed form stationary distributions

  We use continuous-time Markov chain analysis to establish the transition rates and consequently the stationary state distributions used to find the stationary drop probabilities for nonlinear RED and modified nonlinear RED algorithms. Based on reasonable approximations and Markov chain analysis, we derive the stationary distribution of the number of packets between consecutive drops and the stationary distribution of observed queue with closed form equations, and then we derive closed form stationary drop probabilities.

- Derivation of the closed form distribution of consecutive drops

  According to the stationary drop probabilities we derived, we give the closed form distributions of the number of consecutive drops for nonlinear RED and modified nonlinear RED algorithms. Furthermore, we derive the expectation and variance of the number of consecutive drops.

- Derivation of correlation between the stationary average queue occupancy and the offered load

  We derive the stationary average queue occupancy as the function of the offered load with a closed form equation. We can get the value of the stationary average queue occupancy given the offered load for nonlinear RED and modified nonlinear RED algorithms.

- Derivation of correlation between the latency factor and compensation factor

  We give guideline how to choose latency factor and compensation factor. We derive the correlation between the latency factor and compensation factor with the closed form equations.

3. Analyze and evaluate the performance of modified nonlinear RED algorithm and classical linear and nonlinear RED algorithms in different cases.

The results show that our modified non-linear RED algorithm has much lower consecutive dropping probability than classical linear (May's analysis model) and non-linear RED algorithms, so our model can avoid the synchronization of TCP flows more efficiently.

6.2 Future Work

In this thesis a modified nonlinear RED algorithm has been developed. Although our modified RED algorithm has been focused on solving the weakness of RED in global synchronization of TCP flows, some interesting research topics as follows are yet to be investigated in the future.

1. Adjust the values of latency factor d and compensation factor m to get their optimum matched values.

   From our numerical results in chapter 5, we found that when latency factor and compensation factor reach a certain point, the consecutive drops decrease very

slowly. So the latency factor d and compensation factor m should have a pair of optimum matched values.

2. To further investigate the performance of our modified nonlinear RED algorithm in bursty traffic by simulation.

   The modified RED algorithm should have better performance to absorb transient bursts because we introduce latency factor to delay dropping packets. But it is hard to get analytical solutions due to more complicated stationary distributions.

3. Investigate the dynamic performance of the modified nonlinear RED algorithm for different weight factors by establishing ordinary differential equations (ODE). Our current analytic solutions are based on assuming that w_q is small and they do not allow us to investigate the system performance for different weight factors. It is necessary to get ODE model to capture the dynamic performance of RED gateway.

4. To further investigate the performance of our modified nonlinear RED algorithm with feedback oriented traffic like TCP.

   Since it is widely accepted that the Poisson model is not sufficient to characterize the traffic in current Internet, further studies are needed to find better analytical model to analyze and evaluate the performance of the modified RED algorithm with feedback oriented traffic.

5. Improving the modified nonlinear RED algorithm further to reduce queue delay.

From our numerical results in chapter 5, even though the modified nonlinear RED gets lower consecutive drops, it may also have higher queue delay than classical linear and nonlinear RED algorithms.

Bibliography

[ALL00] S. Athuraliya, S. Low, and D. Lapsley. *Random Early Marking.* Proceedings of the 1st COST 263 International Workshop: Quality of Future Internet Services, QofIS 2000, pp. 43-54, Berlin, Germany, March 2000.

[AT99] F. M. Anjum and L. Tassiulas. *Fair Bandwidth Sharing among Adaptive and Non-Adaptive Flows in the Internet.* Proceedings of IEEE INFOCOM 1999, pp. 1412-1420, New York, USA, March 1999.

[BCC98] B. Braden, D. Clark, and J. Crowcroft. *RFC2309: Recommendations on queue management and congestion avoidance in the Internet.* http://www.ietf.org/rfc2309.txt, August 1998.

[BOP94] L. S. Brakmo, S. W. OMalley, and L. L. Peterson. *TCP Vegas: New Techniques for Congestion Detection and Avoidance.* Proceedings of ACM SIGCOMM, pages 24-35, October 1994.

[BT01] T. Bu and D. Towsley. *Fixed Point Approximations for TCP Behavior in an AQM Network.* ACM SIGMETRICS 2001, Cambridge, Massachusetts, USA, 2001.

[CF98] D. Clark and W. Fang. *Explicit Allocation of Best-Effort Packet Delivery Service.* IEEE/ACM Transactions of Networking, vol. 6, no. 4,pp. 362-373, Anchorage, Alaska, USA, 1998.

[FGS01] S. Floyd, R. Gummadi, , and S. Shenker. *Adaptive RED: an algorithm for increasing the robustness of RED's Active Queue Management.* http://www.icir.org/oyd, August 2001.

[FH99] S. Floyd and T. Henderson. *The NewReno Modification to TCPs Fast Recovery Algorithm.* Internet Draft draft-ietf-tcpimpl-newreno-02.txt, February 1999.

[FJ93] S. Floyd and V. Jacobson. *Random Early Detection Gateway in Congestion Avoidance.* IEEE/ACM Transactions on Networking, Vol. 1, no. 3, pp. 397-413, 1993.

[FKSS99a] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. *Blue: A New Class of Active Queue Management Algorithms.* UM CSE-TR-387-99, New York, USA, April 1999.

[FKSS99b] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. *A Self-Configuring RED Gateway.* Proceedings of IEEE INFOCOM 1999, pp. 1320-1328, New York, USA, March 1999.

[FKSS01] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. *Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness.* Proceedings of IEEE INFOCOM 2001, pp. 1520-1529, Anchorage, Alaska, USA, April 2001.

[Flo94] S. Floyd. *TCP and Explicit Congestion Notification.* ACM Computer Communication Review, vol. 24, No. 5, pp.10-23, 1994.

[HMGT01] C. Hollot, V. Misra, W. Gong, and D. Towsley. *A Control Theoretic Analysis of RED.* Proceedings of IEEE INFOCOM 2001, pp. 1510-1519, Anchorage, USA, April 2001.

[Hoe96] J.C. Hoe. *Improving the Start-up Behavior of a Congestion Control Scheme for TCP.* Proceedings of ACM SIGCOMM, August 1996.

[Jac88] V. Jacobson. *Congestion Avoidance and Control.* Proceedings of SIGCOMM 88, Palo Alto, CA, August 1988.

[Jco90] V. Jcobson. *end2end-interest mailing list: Modified TCP Congestion Avoidance Algorithm.* ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt, April 1990.

[KMV01] S. Kohler, M. Menth, and N. Vicari. *Modeling RED with Two Traffic Classes.* the 2nd COST 263 International Workshop: Quality of Future Internet Services, QofIS 2001, Coimbra, Portugal, September 2001.

[KV00] P. Kuusela and J. T. Virtamo. *Modeling RED with Two Traffic Classes.* Proceedings of the 15th Nordic Teletraffic Seminar, NTS 15, Lund, Sweden, 2000.

[LM97] D. Lin and R. Morris. *Dynamics of Random Early Detection.* Proceedings of ACM SIGCOMM 1997, pp. 127-137, Sophia Antipolis, France, September 1997.

[MBB00] M. May, T. Bonald, and J. Bolot. *Analytic Evaluation of RED Performance.* Proceedings of IEEE INFOCOM 2000, Vol. 3, 26-30, March 2000.

[MGT00] V. Misra, W. Gong, and D. Towsley. *A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED.* Proceedings of ACM SIGCOMM 2000, pp. 151-160, Stockholm, Sweden, September 2000.

[MM96] M. Mathis and J. Mahdavi. *Forward Acknowledgement: Refining TCP Congestion Control.* Proceedings of ACM SIGCOMM, August 1996.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. *TCP Selective Acknowledgment Options.* RFC 2018, October 1996.

[OLW99] T. J. Ott, T. V. Lakshman, and L. H. Wong. *SRED: Stabilized RED.* Proceedings of IEEE INFOCOM 1999, pp. 1346-1355, New York, USA, March 1999.

[PB99] S. Peeters and C. Blondia. *A Discrete Time Analysis of Random Early Detection with Responsive Best-Effort Traffic.* Proceedings of the 7th IFIP Working Conference on Performance Modeling and Evaluation of ATM and IP Networks, IFIP ATM and IP 1999, Antwerp, Belgium, June 1999.

[Ste97] W. Steven. *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms.* RFC 2001, January 1997.

[SVL00] V. Sharma, J. Virtamo, and P. Lassila. *Performance Analysis of the Random Early Detection Algorithm.* http://www.tct.hut.fi/tutkimus/com2/publ/redanalysis.pdf, 2000.

[WC91] Z. Wang and J. Crowcroft. *A New Congestion Control Scheme: Slow Start and Search (Tri-S).* Computer Communication Review, 21(1):32-43, January 1991.

[WC92] Z. Wang and J. Crowcroft. *Eliminating Periodic Packet Losses in 4.3-Tahoe BSD TCP Congestion Control Algorithm.* Computer Communication Review, 22(2):9-16, April 1992.

[ZC90] L. Zhang and D. Clark. *Oscillating behavior of network traffic: a case study simulation.* Internetworking: Research and Experience, vol. 1, no.2, pp. 101-12, 1990.

[ZFH99] T. Ziegler, S. Fdida, and U. Hofmann. *RED + Gateway for Identification and Discrimination of Unfriendly Best-Effort Flows in the Internet.* Proceedings of the 7th IFIP Working Conference on Performance Modeling and Evaluation of ATM and IP Networks, IFIP ATM and IP 1999, pp. 27-38, Antwerp, Belgium, June 1999.