Analysis and Design of Active Queue Management for TCP-RED Congestion Control Strategies

By

Nabhan Hamadneh

A thesis submitted for the degree of

Doctor of Philosophy

School of Information Technology,

Murdoch University.

November 2012

Declaration

I declare that this thesis is my own account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary education institution.

Signed:_____

Abstract

This thesis investigates the problems of the Active Queue Management (AQM) techniques for congestion control in TCP networks. Random Early Detection (RED) and the RED-based stratgies, which adopt the AQM approach, are evaluated through simulation. Two main problems of RED, and its variants, are considered. The first problem is the mismach between the average and actual queue sizes. The second problem is the parameter configuration. This thesis proposes three new RED-based strategies and simulates them using the NS-2 simulator. These novel strategies are evaluated and compared with current RED based strategies. The proposed strategies are: Queue Sectors RED (QSRED), Risk Threshold RED (RTRED) and Weighted RED (WTRED). The performance of these strategies is evaluated using performance indicators such as: throughput, link utilization, packet loss and delay.

QSRED divides the router buffer into equal subsectors and monitors the queue dynamics. The actual drop probability p_a and maximum drop probability max_p are adjusted depending on the position of the actual and average queue sizes; q and avg respectively.

Currently, RED maintains a maximum threshold max_{th} and minimum threshold min_{th}. The second RED modification, RTRED, adds a third dropping level. This new dropping level is the risk threshold $risk_{th}$ which works with the actual and average queue sizes to detect the immediate congestion in gateways. Congestion reaction by RTRED is on time. The reaction to congestion is neither too early, to avoid unfair packet losses, nor too late to avoid packet dropping from time-outs.

The third proposed RED modification, WTRED, adjusts the weight parameter w_q dynamically, to reduce the mismatch between the average and ac-

tual queue size. WTRED also adjusts the maximum and minimum thresholds, to increase network throughput and link utilization.

The simulation results demonstrate the shortcomings of RED and RED-based strategies. The results also show that QSRED, RTRED and WTRED achieve greater network performance over other strategies.

Contents

List of Figures

List of Tables

Dedication

"...My Lord, enable me to be grateful for Your favor which you have bestowed upon me and upon my parents and to do righteousness of which You approve. And admit me by Your mercy into (the ranks of) Your righteous servants."[An'naml, 19].

I lovingly dedicate this thesis to my brothers, sisters and my fiancee Sheraz for their love and support. Deepest appreciation to my eldest brother Safwan for his financial support.

Acknowledgement

I owe an immense debt of gratitude to my principle supervisor Dr. David Murray for his encouragment and advice in preparing my publications and writing this thesis. I would like to thank my second supervisor Prof. Michael Dixon who agreed to cooperate; in which supervision of this thesis was greatly needed. I would like also to thank my third supervisor Assoc/Prof. Peter Cole for his support from the earliest stage of my Ph.D course.

Publications

- Nabhan Hamadneh, David Murray, Michael Dixon, and Peter Cole: *Dynamic Weight Parameter for the Random Early Detection (RED) in TCP Networks,* International Journal of New Computer Architectures and their Applications, Vol. 2 No. 2, (2012).

This paper presents a new configuration for the weight parameter in RED. The dynamic weight parameter enhances the performance of RED. This paper is edited in Chapter 5.

- Nabhan Hamadneh, David Murray, Michael Dixon, and Peter Cole: *Weighted RED (WTRED) strategy for TCP congestion control*, Communications in Computer and Information Science, Vol. 252 No. 2 pp. 421-434, (2011).

This paper presents WTRED strategy for congestion handling. The strategy uses adjustable weight parameter to avoid the mismatch between the microscopic and macroscopic behaviours of AQM strategies. This paper is edited in Chapter 5. The paper also won the prize of the best paper in the ICIEIS 2011 conferance, Kuala Lumpur, Malaysia.

- Nabhan Hamadneh, Michael Dixon, Peter Cole, and David Murray: *A third drop level for TCP-RED congestion control strategy*, Proceedings of World Academy of Science, Engineering and Technology, Vol. 81 No. 57 pp. 892-898, (2011).

This paper presents RTRED congestion control strategy. The strategy adds a third threshold to enhance the performance of RED. This paper is edited as Chapter 4.

- Nabhan Hamadneh, David Murray, Michael Dixon, and Peter Cole: *QSRED, An Algorithm To Solve The Mismatch Between The Microscopic and Macroscopic Behavior of RED Gateways,* International Journal of Computer Science and Network Security, Vol. 10 No. 11 pp. 63-70, (2010).

This paper presents QSRED congestion control strategy. The strategy adjusts the algorithm of ERED to enhance the network performance. This paper is edited as Chapter 3.

CHAPTER 1

Introduction

Transmission Control Protocol (TCP) is the transport layer protocol that governs 90% of data transfer in the Internet [10] [81] [32]. As the Internet has evolved and the number of users have dramatically increased, new techniques should be developed to grant fair resource allocation between users.

Congestion is created when demand exceeds the available capacity. Due to uncoordinated resource sharing, the Internet has suffered from the problems of long delays in data delivery, wasted resources due to lost or dropped packets, and even possible congestion collapse, which appears when the entire network stalls [38].

Congestion can be solved by numerous approaches. Traditionally, Internet routers used the First In First Out (FIFO) system, dropping from the tail of queues to control congestion. Queue management has evolved over the years and become an active area of research. This thesis proposes new queue management strategies.

1.1. Overview of the congestion problem

The main data streaming unit in TCP is the byte. TCP assigns a sequence number to each byte transmitted; this is then used for flow control and data acknowledgment. In the Internet, however, data transferred in segments (packets). Upon data segment arrival, the destination acknowledges the receipt of the segment by sending an acknowledgment (ACK) with the next expected data segment number. If the ACK is not received within a time-out interval, the data is retransmitted. The receiver advertises the number of bytes it can receive beyond the last received TCP segment. This prevents overflows in the receiver's internal buffer. The advertised number of bytes is

called the advertised window and is included in the header of each ACK sent to the source.

The Round Trip Time (RTT) is the time spent between sending a segment and its ACK received by the sender. This includes the propagation, transmission, queuing, and processing delays over all intermediate routers.

In TCP networks, segments traverse the queues of routers. A characteristic of TCP traffic is that segments may arrive in bursts from one or more sources. Buffers help routers absorb bursts until they can recover. In the case of excessive traffic, buffers are overloaded, and new incoming packets are dropped. Statistical solutions such as: increasing the buffers size or adding more buffers are ineffective [64] because excessive bufferring can cause lengthy delays [65]. Thus, the congestion window *cwnd* is used to prevent the gateway buffer being overwhelmed.

Network congestion is indicated by two events: the first is the packet loss which is detected by a time out [38]; while the second is the duplicated acknowledgment [67]. In case of congestion, TCP reduces the window size (*cwnd*) which represents the data sending rate. This adjustment depends on the rate of acknowledgments arriving at the source. The TCP source node adjusts the congestion window based on congestion signals. It decreases it when the level of congestion goes up and increases it when the level of congestion goes down. This mechanism is commonly called Additive-Increase/Multiplicative-Decrease (AIMD) [12]. It is clear that the increment/decrement of the ACK arrival depends on the state of intermediate routers. In other words, the ACK's arriving and data sending rates will be matched, unless there is a congested router. This automated TCP congestion detection exists according to three main modes:

- Slow start and congestion avoidance: TCP-Tahoe [38].
- Fast Retransmit and Fast Recovery: TCP-Reno [67] [2].
- TCP-Vegas [8].

These TCP modes and other implementations are further investigated in section 2.6 on page 16.

1.2. Queue management and congestion control

The Internet is a packet delivery system. Congestion can cause high packet losses and increase delays; which reduce network performance. Hence, congestion control is the most essential element of TCP. Any discussion of congestion would naturally involve queuing. A variety of queuing techniques are used for network buffer management. Proper queue management minimizes the number of dropped packets and network congestion, as well as improving network performance. TCP sources detect congestion through duplicate ACKs. Intermediate routers use Active Queue Management (AQM) approaches to detect network congestion. These techniques operate directly on the router buffer by measuring and monitoring the average queue size.

Current internet congestion handling strategies are used to improve performance, generally indicated by high throughput, high link utilization, low loss rate and low end-to-end average delay. Accordingly, many congestion control approaches have been proposed. Congestion can be controlled using strategies applied by the TCP sender. These strategies are source algorithms, for example, Vegas [8] and Tahoe [38]. Another type of strategy applied by intermediate routers is called the network algorithm, such as, Tail Drop [7] and RED [29]. Source algorithms and network algorithms are required to work together to control congestion.

1.2.1. *Congestion recovery*

Classic network routers manage congestion by setting a maximum queue length (threshold). When the queue size exceeds the allowed threshold, all incoming packets are subsequently dropped until buffer space becomes available. This technique is called Tail Drop (TD) as packets are dropped from the tail of the queue. TD was considered an effective congestion control technique until

the detection of two serious drawbacks. These shortcomings are the full queue and lockout; which are described in chapter 5. Tail drop controls congestion reactively in which congestion is detected after the buffer is overloaded. It can also cause global synchronization when all network sources reduce their sending rate simultaneously. This can lead to low link utilization or lockout problem where a few sources monopolize the whole link bandwidth [27].

Global synchronization and lockout problems can be solved by applying a random drop technique in the intermediate router. Whenever a new packet arrives at a congested gateway, a packet is randomly dropped from the queue. Therefore, the probability of dropping a packet from a particular flow is proportional to the bandwidth share of that flow. This technique informs aggressive users that their traffic contributes to congestion more than other users in the network. Random Drop (RD) provides fair resource allocation between connections and improves the network throughput for connections with longer RTTs [33].

1.2.2. *Congestion avoidance*

The congestion recovery techniques react slowly to congestion. These techniques often detect congestion after the buffer has overflowed. They do not solve the full queue problem, and for this reason, the congestion avoidance approach has been proposed.

The full queue phenomenon can be avoided by predicting congestion in its early stages. Thus, a proactive approach must be implemented in the intermediate router to detect the initial stages of congestion. This approach is often referred to as Active Queue Management (AQM). Instead of the actual packet drops, AQM marks packets at the congested router and informs the sources to slow down. The marking technique can be implemented by a bit set in the packet header.

AQM achieves the following goals:

- Prevents global synchronization.
- Determines an average queue size and reduces delays.
- Removes bias against bursty traffic.
- Determines and penalizes aggressive users causing congestion.
- Reduces the packet drop rate using the marking technique.

Random drop was originally proposed as a congestion recovery approach. However, with the Early Random Drop (ERD) enhancement, it can be used as a congestion avoidance technique. The drop rate in ERD is bounded by the level of congestion in the gateway. The basic design of the ERD is to set a fixed threshold. Whenever the queue size exceeds this threshold, packets are dropped randomly from the queue. More sophisticated forms of ERD drop packets based on an Exponentially Weighted Moving Average (EWMA) instead of the actual queue size. ERD outperforms the traditional TD technique in terms of flow segregation, however, the ERD technique does not provide fair resource allocation or restrain aggressive users. ERD does have bias against sudden bursts of traffic [33].

Random Early Detection (RED) was proposed by Floyd and Jacobson [29] to overcome the drawbacks of ERD. The main goal of RED is to provide an efficient mark/drop rate to control the average queue size and avoid biases against bursty traffic. Initial stages of congestion are detected by the average queue size. If the average queue size exceeds a preset threshold, RED drops/marks arriving packets with a drop probability; this is a function of the average queue size. RED gateways maintain low average queue size whereas occasional bursts are allowed to pass through.

RED has been a hugely influential strategy in the area of AQM. It maintains a set of parameters to control congestion. However, RED is very difficult to parametrize, it being possible for RED performance to operate similarly to tail drop under some traffic conditions. Parameter measurement and the recommended values have been subject to change over time. Subsequently, many

RED-based strategies have been proposed since the original RED proposal. In almost all studies, the parametrization of RED depends on simulation observations and examination under particular traffic conditions.

This thesis proposes three RED-based strategies with new parameter configurations to increase network performance.

1.3. Organization of the thesis

Chapter 1 has reviewed the fundamentals of congestion control. Chapter 2 presents the TCP source congestion control strategies and the queue management approaches, with a particular focus on Active Queue Management (AQM). Chapter 2 also describes the Random Early Detection (RED) and RED-based strategies for handling router congestion.

After a thorough review of TCP and AQM; three RED-based strategies along with their performance evaluation are proposed in Chapters 3, 4 and 5. Several problems, associated with the design and implementation of current RED-based strategies, are addressed. The performance of the proposed strategies is compared with RED variants using the NS-2 simulator.

Chapter 3, investigates the mismatch between the microscopic and macroscopic behaviours of RED queues. An active queue management strategy called Queue Sectors RED (QSRED) is proposed to solve this problem. QSRED monitors the queue dynamics in different queue sectors to avoid this mismatch. Chapter 4 presents a novel active queue management algorithm RTRED. Using a third drop level, RTRED reduced the unfair packet drops. Similarly, more disadvantages of current RED-based strategies are described in Chapter 5 and a novel RED-based strategy (WTRED) is proposed to overcome these shortcomings. WTRED uses adjustable weight parameter to reduce the mismatch between the microscopic and macroscopic behaviours of the queue.

CHAPTER 2

Literature Review

This Chapter presents a literature review of the congestion problem. Sections 2.1 to 2.6 summarize the history of TCP congestion control. Traditional TCP variants such as, Tahoe, Vegas and AIMD are described in these sections. The readers can skip sections 2.1 to 2.6 if they are familiar with these topics. Sections 2.7 and 2.8 presents congestion control using the queue management approach in intermediate routers. Section 2.9 describes the Random Early detection (RED). RED-based strategies are described in section 2.10. The problems associated with Active Queue Management (AQM) are described in section 2.11. Section 2.12 describes some non AQM based congestion control techniques. Finally, the Chapter is summarized in section 2.13.

Network congestion is a phenomenon caused by the overloading demand for finite network resources. When demand exceeds the available capacity of the network, performance will be reduced resulting in long delays and higher packet losses. The most severe result of congestion is the collapse of a network, in which the entire network stalls [64].

It is very important to design congestion control strategies that keep the network operating optimally. Congestion control strategies should decrease the demand by adjusting the sending rate to the congested links. Allocating more buffers or providing faster links, which are the features of many statistical solutions, do not prevent the network from congestion [64]. Theses statistical solutions can cause severe congestion resulting in poor performance.

The term flow control is sometimes confused with congestion control. Flow control is a window-based technique used by TCP sources to prevent slow receiver buffers from being overwhelmed by fast senders. Thus, these strategies are called source algorithms. The congestion window (*cwnd*) is the

amount of data that can be sent before receiving an acknowledgment. In the absence of congestion, the window is increased to increase the sending rate. In case of congestion, the window is decreased to reduce the sending rate until congestion is reduced.

TCP flow control techniques, such as TCP-Tahoe and TCP-Reno, have been in use for long period of time. However, buffer space in the routers is finite, and therefore techniques are required to manage the queue size. Hence, various congestion control strategies; such as Random Early Detection (RED), have been proposed to prevent congestion in the intermediate router. This type of strategy is a network strategy because it is applied by network components, such as routers, to control congestion.

Accordingly, there are two approaches to deal with network congestion; reactive and proactive approaches. The reactive approach, which is applied by source algorithms such as Tahoe and Reno, starts recovering after a network buffer is overflowed. The proactive approach aims to prevent buffer overflow using strategies applied in the intermediate router. RED [64] is an example of a proactive approach.

2.1. What is congestion?

It is normal for several IP packets to arrive simultaneously at the intermediate router; waiting to be forwarded over a congested link. Hence, intermediate routers use buffer space to queue packets until they can be serviced. In some stages, the sending rate of network sources exceeds the available buffer space in routers and a packet has to be dropped. In this case, traditional routers, which use a First In First Out (FIFO) queue management technique, drop packets from the tail of the queue.

An infinite buffer would not solve congestion; because the generated queue will be unlimited which increases the end-to-end delay. A packet exposed to long delays may have already timed out and been retransmitted by

the source [55]. A small buffer size is better than a large one; because this will reduce delays and save memory resources.

2.1.1. *Congestion collapse*

Congestion increases delays in data delivery and wastes network resources through lost packets. It could also lead to congestion collapse when all network connections cease. Congestion collapse dramatically degrades network throughput as illustrated in Fig. 2.1. Congestion collapse has appeared in different forms since the early days of the Internet. The first instance was described in 1984 [54]. Nagle noticed duplicated retransmissions for packets already received or still in transit. This form of congestion collapse reduced network throughputs.

The severe form of congestion collapse is caused by undelivered packets. This form of congestion collapse wastes the network bandwidth, due to dropping packets, before they reach the end destination. The solution to this form of congestion collapse is to reduce the offered load. There are also two forms of congestion collapse described by Sally Floyd [23], which are fragmentation-based congestion collapse and congestion collapse from stale packets. The former is a result of transmitting packet fragments that will be definitely discarded by the receiver, because they cannot be reassembled into a valid packet. The latter caused through wasting the network bandwidth with packet transmissions, which are no longer wanted by the user.

2.1.2. *Congestion and misbehaving users*

Misbehaving users ignore congestion signals and tend to use more than their allowed bandwidth. Thus, misbehaving users receive better service than cooperating users and degrade the stability and operation of the network.

TCP traffic is responsive traffic, which reduces the sending rate in response to congestion signals. TCP is designed to share the network with other

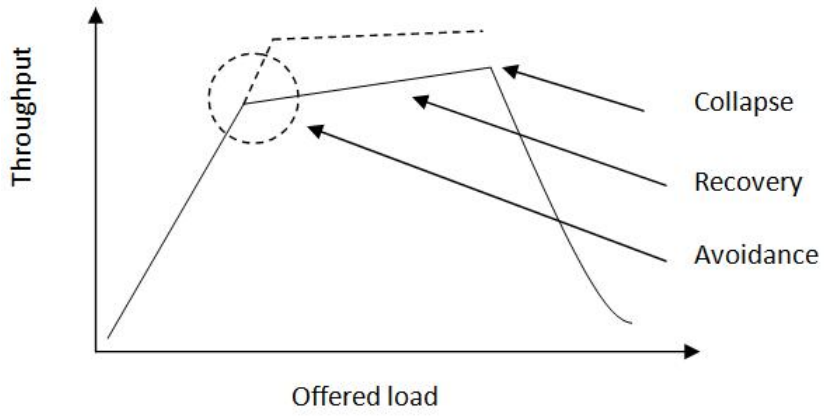


FIGURE 2.1. Throughput as a function of the offered load.

types of traffic, such as UDP, which is unresponsive to congestion signals. In the absence of proper congestion control techniques; misbehaving UDP traffic users will use more than their fair share of network resources such as bandwidth. Therefore, it has been suggested [54] that misbehaving users should be disconnected. It has been also suggested by Floyd in [23] that the best place to detect misbehaving users is in the network itself.

Stability is measured by the level of cooperation in the network [4] because it allows for fair resource allocation.

2.1.3. *Fair resource allocation*

Fairness becomes a problem when users compete for their resource share with unsatisfied demands. Fair resource allocation is satisfied when each user's throughput matches all other users that share the same bottleneck link [39]. Hence, the Internet Engineering Task Force (IETF) has considered the operation of fairness as follows [58]:

- Resources are allocated in increasing order of demand.
- A user is never allocated a share higher than its demand.
- All users with unsatisfied demands are allocated equal shares.

2.1.4. *Congestion Control approaches*

Resource allocation consists of two parts; end-to-end and per-link management [60]. Therefore, congestion control is classified as host-based and router-based techniques; relating to the place in which these techniques are implemented.

Flow control is implemented in the end hosts. Network sources are responsible for end-to-end flow control depending on routers feedback. When the gateway is about to be overloaded, TCP sources reduce the sending rate until the gateway recovers from congestion. Feedback can be implemented by the end host to avoid local buffer overflow in the case that a fast sender is overwhelming a slow receiver buffer. However, feedback can be implemented separately by intermediate routers; in which packet drops are the main signal of congestion. TCP responds implicitly to packet drops by reducing the sending rate, which is the simplest form of congestion control. However, treating the router as a black box has some limitations on resource allocation control and reduces the services provided by the network.

Routers are aware of the initial stages of congestion; thus they have to play a greater role in congestion control. There are two techniques used by routers for buffer and bandwidth management. The first option is to make direct bandwidth allocation on an output link. The second option is to make indirect bandwidth allocation by managing the router queue size.

There are two approaches for congestion control: the first is the congestion recovery which is often confused with the term congestion control. The second approach is congestion avoidance. Congestion recovery approach starts controlling congestion after the gateway is overloaded. Conversely, congestion avoidance applies measures to reduce congestion before the gateway is overloaded. In the literature, the term congestion control is used to denote both approaches.

2.2. TCP performance parameters

The literature has used different parameters, with different definitions, to measure the performance of TCP networks. Five parameters are commonly used for this purpose [25] [16] [1] [29] which are: throughput, link utilization, packet loss, delay and jitter. Following are some of the other parameters that are used to measure network performance [34]:

Round trip delay: The time required for an IP datagram to travel from source to destination and back. This parameter includes propagation time and queuing delay.

One way delay: Due to network characteristics such as routing and bandwidth limitations; the one way delay is not always half of the round trip delay. The estimation of this parameter is very important in highly interactive applications, such as voice over IP.

Maximum delay: It represents the maximum allowed one way delay for high network performance rate.

Delay jitter: Reflects the variability of one way delay between IP datagrams. Higher jitter indicates lower network performance, especially for multimedia applications.

Packet loss rate: Packets can be dropped by intermediate routers due to buffer overflow. They can be also lost at buffers due to physical errors. The loss rate is the ratio of correctly received packets to the number of propagated packets.

Effective bandwidth: It is also called the effective throughput which is the number of bytes that transferred in one second through the network.

Throughput variation: The variability of throughput over a given time scale.

File transfer time: The time taken by a file or object to be transmitted from source to destination. This can be estimated using the

connection effective throughput.

Fairness: This parameter reflects the fairness of resource allocation between network clients.

Resource consumption: TCP techniques that consume fewer network resources reflect high network performance.

2.3. Acknowledgments

Four types of acknowledgments are used to confirm data delivery at the destination. These acknowledgments are [34]:

ACK-only segment and piggybacking: The receiver sends ACKs packet by packet. The ACK can contain a zero payload; which is called the ACK-only technique. Another option to include the ACK number within a data packet, is called the Piggybacking.

Delayed ACK: Instead of acknowledging packets one by one; a delayed acknowledgment can be sent for a group of segments.

Duplicate ACK: A duplicate ACK can be used to indicate missing packets implicitly in TCP networks. For example, if $n-1$, n and $n+1$ are subsequent packets. When packet number $n-1$ is received, the receiver sends ACK number n. This is to inform the reciever that packet number $n-1$ is received safely and the next expected packet is number n. If packet number n is lost in the network and packet number $n+1$ is arrived then it is implicitly known by the receiver that packet number n is missing. In this case, the receiver sends ACK number $n+2$ to confirm the recipient of packet number $n+1$ and to inform the sender of the next expected packet which is $n+2$. Even though the ACK number n has been previously triggered to the sender, a duplicated ACK with number n is resent to the sender to reorder segment number n. Some TCP variants take benefits out of this technique to control congestion.

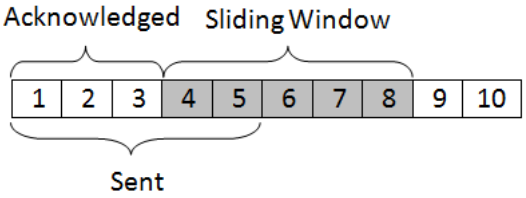


FIGURE 2.2. Example of a sliding window in TCP networks.

2.4. Flow control and sliding window

Time out or triple acknowledgment are used as signals of congestion in TCP networks. When congestion occurs, a TCP congestion control algorithm informs the TCP sender to reduce the sending rate, by halving the congestion window size. The self-clocking behavior is a method used by network sources to regulate the sending rate. Section 2.5 describes this behaviour. The maximum number of bytes that can be released unacknowledged by TCP senders is called the window size. As this window increases and decreases to regulate bandwidth, it is known as the sliding window and is shown in Fig. 2.2.

In this example, the window size is five bytes and the bytes numbered 1 to 5 have been sent. Technically, the next window would contain bytes 6 to 10, but the actual operation is slightly more complex. Since bytes 1 to 3 have been acknowledged but 4 and 5 have not been acknowledged yet, they would not be excluded from the next window; unless they are acknowledged. Instead of that, the next window would contain bytes 4 and 5 before it slides to cover the bytes 6, 7 and 8. This technique is helpful in case bytes 4 and 5 are lost and have to be retransmitted by the sender.

2.5. Traditional source congestion control

Flow control techniques in TCP are used to dynamically adjust the senders advertised window. The available buffer space at the receiver is the criteria to do this adjustment. Due to the design principles of TCP, intermediate routers would not send acknowledgments to the sender. It is, also, not permitted for them to adjust the congestion window *cnwd*. Thus, another

set of mechanisms are applied to indicate congestion at intermediate routers. These mechanisms depend on packet loss or time out to adjust the congestion window. The actual transmission window is set to the minimum of the advertised window and congestion window. The following sections describe three of the traditional source techniques that are used to control congestion in intermediate routers.

2.5.1. *Slow start*

Slow start mechanism requires a sender to start transmissions slowly and then increase the sending rate. The maximum number of packets that can be sent is not predetermined. Hence, the Slow Start mechanism transmits packets slowly to probe the capacity of the network. The sender starts by sending one packet. When the ACK returns, the sender increases the window size by one. Window growth is exponential during this phase [65]. When ACKs stop arriving, the sender determines that it has reached the network capacity. Consequently, the window is degraded to one packet which takes the sender back to the slow start phase.

Slow start is not a congestion prevention technique. It smooths out the sending rate to prevent immediate congestion. Gateway overloading is unavoidable, and eventually, a segment has to be dropped. Therefore, this technique would not be an efficient way to avoid congestion collapse or to avoid long delays in data delivery [38].

2.5.2. *Congestion avoidance*

As an effort to prevent gateway overloading, this technique bounds the exponential increase of the slow-start congestion window. When the window size reaches a threshold that is called the slow start threshold (*ssthresh*) the window is increased linearly, one packet per ACK. Fig. 2.3 illustrates this technique.

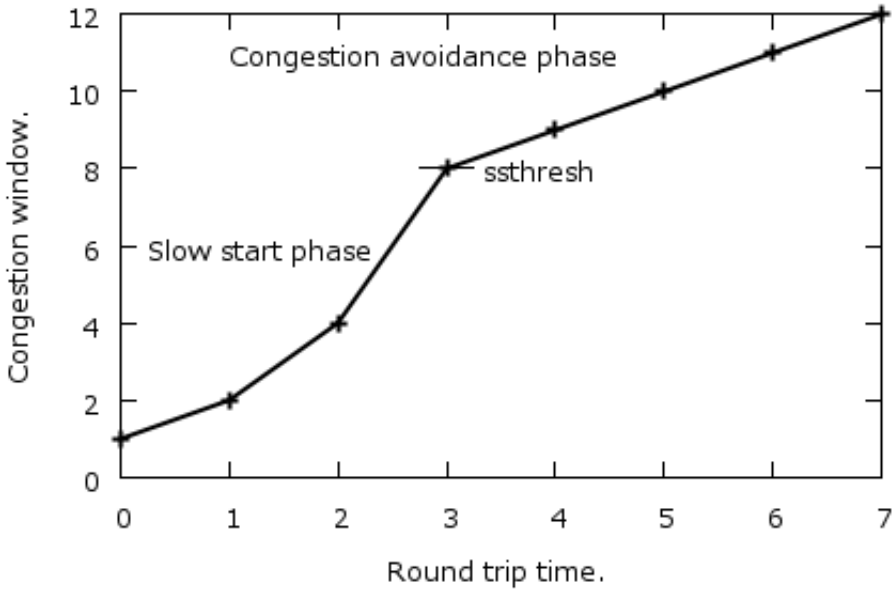


FIGURE 2.3. Slow start and congestion avoidance.

2.5.3. *Additive-Increase/Multiplicative-Decrease(AIMD)*

The classic TCP variants used to maintain prefixed *cwnd* and *ssthresh* parameters. Newer variants of TCP, such as Reno, adjust these two parameters dynamically. At the congestion avoidance phase, the *cwnd* parameter is increased by fixed amount of data, normally one packet. This increment is called the Additive Increase. Time outs or duplicate Acks halves the *ssthresh* and reduces the *cwnd*, normally to one packet, returning the sender back to the slow start phase. These two decrements of the *cwnd* and *ssthresh* are called the Multiplicative Decrease. In case of successive time out events, the parameter *ssthresh* will be reduced exponentially until it reaches the value two.

2.6. TCP variants

Some drawbacks were discovered in the congestion control algorithms of the classic TCP implementations. As discussed in previous sections, congestion collapse [41] was one of the serious troubles caused by these shortcomings. Therefore, many TCP modifications, such as TCP-Tahoe [38], were proposed

to overcome these problems. The following section reviews the most common TCP variants:

2.6.1. *TCP-Tahoe*

A time out event in traditional TCP variants is interpreted as a packet lost causing the sender to retransmit lost packets. TCP-Tahoe uses different techniques to retransmit the lost packet [38]. Duplicate ACKs are used to speed up packet retransmission before the timer expires. TCP-Tahoe reacts to duplicate ACKs by reducing the *ssthresh* parameter to half of the current *cwnd* and the *cwnd* itself is reduced to one packet; which means that the network has entered the slow start phase. As the traffic is accumulated and the sending rate is heavily increased, the network enters the congestion avoidance phase.

2.6.2. *TCP-Reno*

TCP-Tahoe and TCP-Reno [67] [2] use duplicate ACKs to indicate packet loss. Instead of returning to the slow start phase in TCP-Tahoe, TCP-Reno goes into the congestion avoidance phase by halving the parameter *cwnd* and assigning the new *cwnd* to the parameter *ssthresh*; ($cwnd = ssthresh$). This new phase of TCP-Reno is called the fast recovery phase. TCP-Reno keeps track of all lost packets and tries to retransmit them during this phase. TCP-Reno does not enter the slow start phase unless the congestion window becomes very small due to multiple packet losses.

2.6.3. *TCP-Newreno*

TCP NewReno does not leave the fast recovery phase due to multiple packet losses [13] [26] [36]. It reduces the congestion window by the number of acknowledged packets minus one, then assumes that the packet after the most recently acknowledged one is lost and retransmit it.

2.6.4. *TCP-SACK*

Selective ACK (SACK) is an extension that can be enabled alongside NewReno. SACK is sent to the sender when multiple packets are lost. The sender uses this SACK as an image of the receiver queue, then detects and retransmits the lost packets without waiting for a time out [13] [49]. New packets cannot be sent unless all outstanding packets are acknowledged. If ACKs are not received on time, TCP-SACK uses Reno's time out technique to retransmit lost packets.

During the fast recovery phase, a parameter called *pipe* is set to the number of outstanding packets in transit. The sending rate should be less than the *cwnd*. Therefore, the sender sends data only when *the pipe* value is less than the *cwnd*. The *pipe* value is increased by one with every sent packet and decreased by one with every ACK included in a SACK. The *pipe* is reduced by two only with multiple packet losses.

2.6.5. *TCP-FACK*

TCP Forward Acknowledgment (FACK) is designed to retransmit multiple lost packets. This technique maintains information about the highest sequence number of acknowledged packets using two parameters which are $fack$ and $retran - data$. The first maintains the sequence number of the last confirmed packet by SACK. The second, maintains the number of retransmitted but not yet confirmed packets. The amount of outstanding data is estimated using Eq. 2.1.

$$OSD = EFDS - fack + retran - data \tag{2.1}$$

Where:

OSD: The amount of outstanding data.

$EFDS$: Estimated forward data sent.

FACK does not change the value of the *cwnd* during the fast recovery phase. Instead, it keeps the rate of outstanding data to be one segment within a *cwnd*. Thus, the fast retransmit in TCP-FACK is more promptly [48].

2.6.6. *TCP-Vegas*

Vegas [8] is a TCP variant that does not reduce the congestion window relating to segment loss. Vegas pipe is assigned the value of the expected throughput. When the network becomes congested, the actual throughput will be less than the expected throughput and the *cwnd* is adjusted relating to this event.

The sending time of each segment is recorded and the related round trip time is estimated upon ACK arrival. A base round trip time ($BaseRTT$) is maintained for each connection. This parameter is assigned the value of the lowest recorded round trip time among the packets that have been sent through the same connection. The expected throughput is estimated using Eq. 2.2.

$$Ex_{th} = WZ/BaseRTT \tag{2.2}$$

Where:

Ex_{th}: The expected throughput.

WZ: The current congestion window size.

The actual throughput is calculated per round trip time. Fig. 2.4 illustrates the algorithm of Vegas, where $Diff$ is the difference between the actual and the expected throughput. Two parameters, α and β are normally used to represent the heavy weight and the light weight of network data respectively.

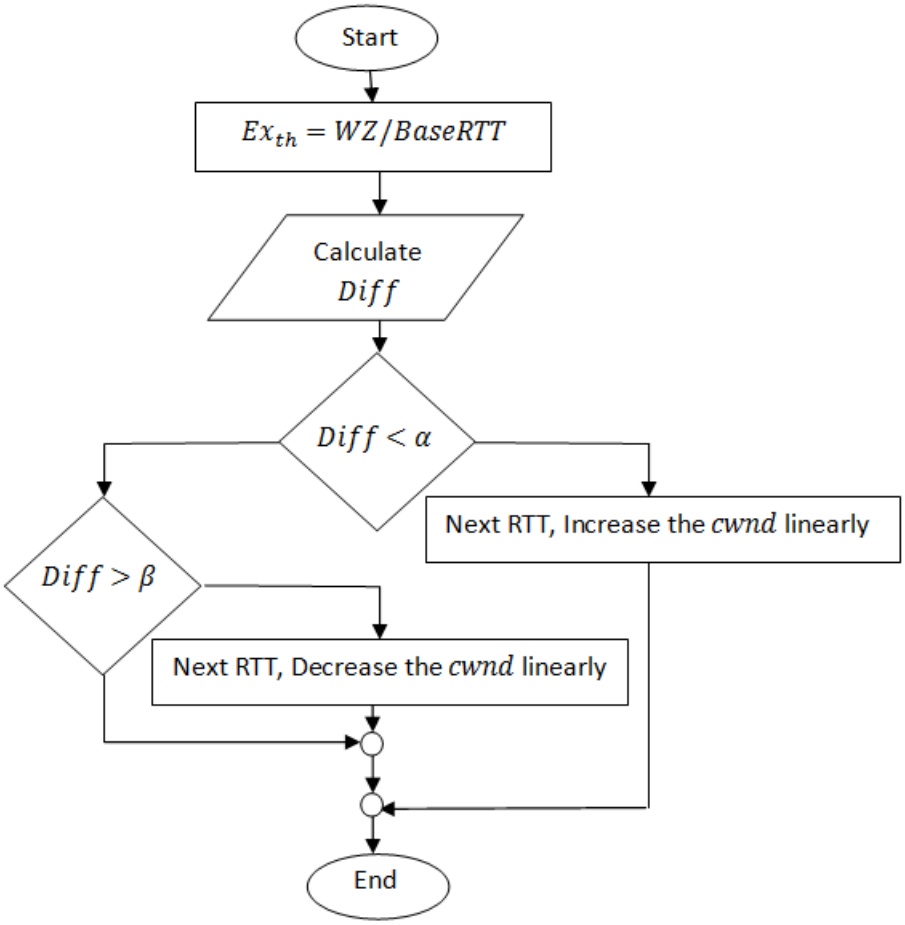


FIGURE 2.4. TCP Vegas algorithm.

2.6.7. *TCP Vegas modifications*

A modification of slow start independent from packet loss was proposed with TCP-Vegas. The congestion window is adjusted exponentially every second round trip time and the other round trip time is used to calculate the $Diff$ parameter. If $Diff$ is less than a preset parameter α then Vegas jumps from the slow start phase to the congestion avoidance phase. This provides better estimation of the available bandwidth of the connection.

As a new retransmission technique, Vegas retransmits packets for each duplicate ACK, instead of waiting three duplicate ACKs. This new technique is only applicable if the estimated RTT is greater than the time out value. If lots of packets are lost in the same block, or if the window size is small, then the triple ACK becomes an impossible condition. This problem can be fixed

using the modified Vegas.

2.7. Queue management and congestion control

TCP networks apply queue management algorithms to control congestion in intermediate routers. There are two main approaches to control congestion. The first approach is the Passive Queue Management (PQM). The second is the Active Queue Management (AQM). PQM is a congestion recovery approach which starts congestion control after the gateway buffer is overloaded. AQM is a congestion prevention approach in which congestion control starts before the gateway buffer is overload.

In PQM, if a packet is dropped at the intermediate router due to buffer overflow, then senders have to wait for time out event before they reduce their sending rate. In this case, congestion is detected by senders implicitly. This reaction by senders is called the Implicit Congestion Notification (ICN). In contrast, the Explicit Congestion Notification (ECN) is a technique that allows intermediate routers to mark the congestion bit in a packet header rather than dropping it. When ECN marked packet arrives at the destination, the receiver sets the congestion bit in the ACK's header before it is replayed to the sender. ECN is very effective with moderate cases of congestion. If congestion is excessive then dropping packets is considered to be more efficient solution [62].

Normally, PQM threshold is the buffer limit, but AQM threshold is sufficiently less than the buffer limit. This makes a difference between AQM and PQM approaches. In PQM, when the queue size reaches a prefixed threshold, all incoming packets are dropped in order. Conversely, AQM drops packets randomly.

A lot of strategies have adopted the PQM approach. Drop-From-Front (DFF) is a PQM strategy that drops packets from the front of the queue to make space for recently arrived packets. Push-Out (PU) is another implementation of PQM in which the packets are dropped from the end of the queue

to make space for the new arrived packets. The well known Tail Drop (TD) strategy drops all arriving packets from the tail of the queue. Despite the benefits of queue management and congestion handling, PQM suffers from some problems. Full Queue problems occur when the gateway continually sends full queue signals to sources for a long period of time. Lock out occurs when TD allows a few connections to monopolize the whole buffer space. Global Synchronization occurs when all TCP senders reduce their sending rate simultaneously; resulting in low network throughput. These problems are further described in subsequent chapters.

Active Queue Management (AQM) strategies, such as Random Early Detection RED [29], were proposed to solve the problems of the Passive Queue Management (PQM) strategies. AQM improves network throughput by reducing the number of dropped packets in the intermediate router. Moreover, the small queue size which is maintained by AQM reduces the average network delay [34].

2.8. Intermediate router congestion control

Prior to the introducton of DECbit in 1988, routers maintained large buffers to accommodate transient congestion caused by high speed networks. These large buffers resulted in large delays. Thus, it was necessary to minimize the queue size without degrading throughput.

Instead of using packet loss to indicate congestion, it was proposed to detect congestion using other indicators. These indicators include: the estimated bottleneck link service time, changes in throughput as well as changes in end-to-end delay. However, it has been shown that the most efficient place to detect congestion is in the intermediate router itself [29].

DECbit [40] is one of the earliest congestion control strategies that was applied to intermediate routers. When the average queue size exceeds a prefixed threshold, DECbit informs senders of congestion explicitly. It sets a congestion bit in the packet header to inform senders of congestion. For every

packet arrival, the average queue size is calculated for the current gateway busy period. Also, the overhead of the previous busy and idle periods will affect the calculation of the current average queue size. If the calculated average is more than one, the congestion bit is set in the packet header. For every second round trip time, if half or more of the packets from the last window were marked then the source reduces its sending window size exponentially. Otherwise, the window is increased linearly.

Early Random Drop (ERD) [33] is an intermediate router strategy. When the queue size exceeds a predefined level, packets are dropped with a fixed drop probability. It was suggested as a future work in ERD's proposal to maintain adjustable threshold and drop probability [33]. The subsequent sections shows that this enhancement was achieved by the Random Early Detection (RED) [29].

Research shows that ERD solves the global synchronization problems caused by TD [33]. However, ERD is unable to control misbehaving users [78]. This might enable misbehaving users to be assigned an unfair proportion of the bandwidth.

IPSource Quench [61] is a congestion control strategy that revealed the use of two dropping levels or thresholds. The gateway sends a source quench message when the average queue size exceeds the first threshold [60]. If the queue size exceeds the second threshold, the gateway could discard arriving packets other than ICMP packets.

2.9. Random Early Detection (RED)

Random Early Detection (RED) [29] is a gateway congestion control strategy that adopts the active queue management approach. RED maintains two dropping levels and uses an Exponentially Weighted Moving Average (EWMA) to control congestion. This section evaluates the design and performance of RED.

2.9.1. *RED design guidelines*

RED controls the average queue size to avoid congestion in intermediate routers. It was designed to avoid global synchronization and bias against bursty traffic. It also provides techniques to bound the average queue size without transport layer support [29].

The main goal of designing new congestion avoidance techniques is to maintain low delay, high throughput, as well as to detect the initial stages of congestion. Congestion avoidance strategies should also reduce queue sizes to reduce delays. Some queue fluctuation should be permitted to accommodate bursty traffic and transient congestion [64].

As the gateway has direct control over the queue, congestion is easily detected in the gateway itself. This gives the gateway the ability to warn individual senders that are overwhelming the buffer.

Connections are set with different round trip times, throughput requirements and delay sensitivity. The average queue size in RED is estimated using a low pass filter. The gateway is able to determine the appropriate short-lived bursts that can be accommodated by the queue. RED achieves this by controlling the time constants used by the low-pass filter. Thus, RED is able to detect congestion that lasts multiple round trip times [29].

Congestion avoidance schemes must decide which connections are to be notified of congestion. If congestion is detected and the gateway is not yet full then the packet drop technique is an inappropriate option to notify sources of initial congestion. Instead, RED marks packets and the corresponding source reduces its window size. Legacy TCP variants do not provide a congestion bit in packet header. In this case, the packet must be dropped to inform the source of congestion [29].

If many connections were simultaneously notified of congestion via packet loss, they will reduce their sending rate simultaneously; resulting in low throughput [30] [79]. This problem, which exists in Tail Drop gateways, has been

alleviated using RED because RED drops packets randomly from the queue.

Tail Drop and Early Random Drop gateways suffer from bias against bursty traffic [28]. When a particular burst connection is overwhelming the gateway, eventually a packet has to be dropped due to buffer overflow. RED also addressed bias against bursty traffic.

2.9.2. *RED algorithm*

RED maintains an Exponentially Weighted Moving Average (EWMA) of the buffer size on Internet routers [29]. Equations 2.3, 2.4 and 2.5 illustrate how the drop rate of packets is calculated. Table 2.1 illustrates the RED algorithm.

$$avg = (1 - w_q) * avg + w_q * q \tag{2.3}$$

$$p_b = max_p(\frac{avg - min_{th}}{max_{th} - min_{th}}) \tag{2.4}$$

$$p_a = p_b(\frac{1}{1 - count * p_b}) \tag{2.5}$$

Where:

avg : average queue size.

w_q : a weight parameter, $0 \leq w_q \leq 1$.

q : current queue size.

p_b : immediately marking probability.

max_p : maximum value of p_b.

min_{th} : minimum threshold.

max_{th} : minimum threshold.

p_a : accumulative drop probability.

$count$: number of packets since the last marked packet.

TABLE 2.1. Detailed Algorithm of RED.

$Initialization:$
$\quad avg = 0$
$\quad count = -1$
$for\ each\ packet\ arrival$
$calculate\ the\ new\ average\ queue\ size\ avg:$
$\quad if\ the\ queue\ is\ nonempty$
$\quad avg = (1 - w_q)avg + w_q$
$else$
$\quad m = f(time - q_{time})$
$\quad avg = (1 - w_q)^m avg$
$if min_{th} \leq avg \leq max_{th}$
$\quad increment\ count$
$\quad calculate\ probability\ p_a:$
$\quad p_b = max_p(avg - min_{th})/(max_{th} - min_{th})$
$\quad p_a = p_b/(1 - count. * p)$
$with\ probability\ p_a:$
$\quad mark\ the\ arriving\ packet$
$\quad count = 0$
$else\ if\ max_{th} \leq avg$
$\quad mark\ the\ arriving\ packet$
$\quad count = 0$
$\quad else\ count = -1$
$when\ queue\ becomes\ empty$
$q_{time} = time$

$time$:current time.

q_{time} : start of the queue idle time.

$f(x)$: a linear function of the time t.

The RED gateway has two preset thresholds, the maximum and the minimum thresholds. Every time a new packet arrives at the gateway, the avg value is calculated. If this value is greater than the maximum threshold, then all incoming packets must be marked or dropped. If it is less than the minimum threshold, the arriving packet enters the queue without marking or dropping. When this avg value is in between the minimum and the maximum thresholds, then incoming packets will be dropped or marked with probability p_a [22].

In this way, RED achieves the following:

- Connections with higher input rates receive proportionally more drops or marks than connections with lower input rate
- Maintains an equal rate allocation
- Removes biases against bursty traffic
- Eliminates global synchronization

For more details of the RED design principles see [20] [24].

2.9.3. *Calculating RED's average queue size*

A low pass filter is used to estimate the average queue size in RED. The use of this low pass filter prevents transient congestion and short-term bursty traffic from affecting the average queue size. The time constant of the low pass filter is represented by the parameter w_q in Eq. 2.3 which is the weight parameter used to calculate the average queue size. This parameter can be assigned a value between 0 and 1. The recommended default value is 2×10^{-3} [29]. The upper and lower bounds of this parameter are described in the next section.

2.9.4. *Bounds for the weight parameter*

RED gateways do not filter transient congestion if the value of w_q is very large [29]. The weight parameter is bounded by the desirable average queue size. The minimum threshold and maximum threshold are also taken into account when estimating the average queue size. Chapter 5 proposes a new parameter configuration for the weight, minimum threshold and maximum threshold parameters.

Eq. 2.6 is used to estimate the desirable average queue size. This average represents the increase of an empty queue from zero to L packets over L packet arrivals. For example, if the queue has increased from 0 to 100 packets and the desirable average, avg_d, is 4.88 then the weight parameter to achieve this average is 0.001. The minimum threshold must be always greater than the

avg_d. If the weight parameter is set too low then it would not reflect the changes of the average queue size; delaying the initial detection of congestion [29].

$$avg_d = L + 1 + \frac{(1 - w_q)^{L+1} - 1}{w_q} \tag{2.6}$$

Floyd and Jacbson assumed that if a packet arrives at an empty queue with avg equals zero then the desirable average queue size is 0.63 [29].

During steady state, the packet arrival rate will be equal to the departure rate. Thus, for the previous assumption, the actual queue size will be one packet as long as the gateway is in the steady state. Now, the question is how many packets are required to increase the average from zero to 0.63. RED, uses the formula $-1/\ln(1 - w_q)$ to estimate this number [77]. For $w_q = 0.001$, it takes the gateway 1000 packet arrivals to increase the average from zero to 0.63. With $w_q = 0.002$, the number of packet arrivals is 500. For $w_q = 0.003$ this number is reduced to 333 packet arrivals. Thus, it is again the desirable average that determines the lower bound of the weight parameter.

2.9.5. *Configuration of the maximum and the minimum thresholds*

The optimal values of the minimum and maximum thresholds also depend on the desirable average queue size. In order to increase the link utilization, the minimum and maximum thresholds should be sufficiently large, especially, for bursty traffic [29]. The maximum threshold parameter should be based on the maximum average delay permitted by the gateway. It is recommended to set the difference between the maximum and minimum thresholds larger than the typical increase in the calculated average queue size. It is also recommended to set the maximum threshold to at least twice the minimum threshold. The strategy Weighted RED (WTRED), which is presented in Chapter 5, suggests an alternative configuration of these parameters.

2.9.6. *Calculating the packet marking probability*

The initial mark probability p_b is a linear function of the average queue size which is illustrated in Eq. 2.4. This probability determines the number of packet arrivals between two marked packets. The parameter max_p in Eq. 2.4 represents the maximum value for p_b that is allowed before the average queue size reaches the maximum threshold. The RED proposal [29] recommends packet marking at regular intervals. If many marked packets have short or long intervals between them then this will cause global synchronization. Hence, RED uses a uniform random variable to determine the gaps between two marked packets [29].

2.9.7. *Evaluation of RED*

RED has been evaluated against the following aspects [29]:

Congestion avoidance: If RED uses the packet drop mode to indicate congestion then RED grants average queue size less than the maximum threshold. If the packet mark mode is used then RED will use ECN to mark packets. In addition, the average queue size can be effectively controlled if RED uses the optimal weight parameter.

Appropriate time scales: When using ECN, it takes at least one round trip time for the gateway to recognize the reduction in the arrival rate. In RED, the time scales for congestion detection and source to respond are roughly matched. Connections would not reduce the window size as a result of transient congestion at RED gateways.

No global synchronization: Packet marking rate in RED is bounded by the level of congestion. The probability of marking packets is increased with the congestion level. Thus, RED avoids global synchronization by marking packets at low levels of congestion.

Simplicity: Due to the simplicity of the algorithm, RED can be implemented with moderate overhead in current networks.

Maximizing global power: Network power is the ratio of throughput to delay. RED using the marking technique has full control over the average queue size. It was shown that the global power in RED is higher than TD with high link utilization.

Fairness: In contrast to Drop Tail, RED does not discriminate against particular connections [28], because connections receive marking signals proportional to their share of the bandwidth. However, RED does not grant equal throughput. Also, misbehaving users are not fully controlled. The level of congestion is identified easily by RED. Connections that use large bandwidth are also identified. Therefore, these connections can be controlled to organize the throughput during congestion.

RED is appropriate for a wide range of environments: RED drops packets randomly from the queue. This makes RED applicable for connections with different round trip times, throughput and number of active connections. The average queue size reflects the traffic load allowing the marking rate to be adjusted correspondingly. Even with mixed networks (TCP and UDP), RED is able to control congestion by dropping packets. If packet dropping is not recognized by the transport layer in sources then congestion, eventually, will be detected when the average queue size hits the maximum threshold.

2.9.8. *Parameter sensitivity*

The only parameter that must be set in TD is the buffer size. RED maintains a set of parameters that work together to control the average queue size. Also, to control the time intervals to compute the average queue size and maximum rate of packet marking. Congestion avoidance mechanisms should have parameters that are less sensitive and they should be applicable for a wide range of networks with different bandwidths.

The parameters w_q, min_{th} and max_{th} are set according to the desirable queue size and the duration of bursts allowed in the queue. The parameter max_p is the maximum bound of the actual drop probability p_b which has a wide range of settings. In the presence of moderate congestion, RED marks one packet out of $1/max_p$ packets. When congestion becomes heavy, the marking technique cannot control the average queue size. When the average queue size reaches the maximum threshold, RED drops every arriving packet until congestion is controlled.

Parameter configuration in RED is a complex procedure. However, there are few tips in how to configure RED's parameters to increase network performance under a wide range of traffic conditions. Adequate calculation of the average queue size requires the weight parameter to be set greater than or equal to 0.001. Weight parameters set below 0.001 will obstruct RED from recognizing the actual changes in the queue size [29].

Large minimum threshold maximizes the network power parameter, which is the ratio of throughput to delay. The optimal minimum and maximum thresholds is an area that requires further research. Due to the bursty nature of network traffic; fluctuations in network queues becomes a common behaviour. If the minimum threshold is set too low then the average queue size will be too low as well, resulting in poor link utilization.

In order to avoid global synchronization, the difference between the minimum and maximum thresholds must be large enough to avoid marking too many packets at the same time. The difference should be larger than the typical increase in the average queue size during a round trip time. It is suggested by the RED proposal [29] to set the maximum threshold twice the minimum threshold.

2.10. RED-based strategies

RED was proposed by Sally Floyd and Van Jacobson in 1993 [29]. Since then, many RED-based strategies were proposed to increase network perfor-

mance. Some variants propose new parameter configuration for the original RED proposal. Other strategies add extra parameters to enhance RED's functionality.

2.10.1. *Blue RED*

RED relies on the average queue size to detect congestion. Blue-RED [16], uses the history of packet loss and link utilization to manage congestion. In Blue-RED, arriving packets are marked or dropped using a single probability. In case of buffer overflow and continuous packet drops, Blue RED increases the sending rate of congestion notification by increasing the marking probability. In case of empty queue or idle link, the marking probability is decreased. For optimal operation of RED, it must be configured with sufficient buffer space to accommodate bursty traffic. Also, the congestion notification rate must suppress the transmission rate without reducing the link utilization. When the number of active connections is large, the generated traffic is extremely bursty [15] [17]. Bursty traffic causes rapid increases and decreases in the active queue size which defeat RED's reaction [16].

It is recommended, for ideal RED performance, to set the buffer space in the intermediate router twice the delay bandwidth product [70]. Large bandwidth delay products, considerably, increase the end-to-end delay and jitter which affect the running of the interactive applications.

Comparing with RED, Blue-RED maintains a single drop probability p_m which is increased when the gateway buffer is overflowed and decreased when the queue is empty or the link is idle. Table 2.2 illustrates the Blue-RED algorithm.

A variant of Blue-RED updates the drop probability when the queue size exceeds a certain value L. This adjustment provides more free space in the buffer for transient bursts of traffic. It also allows the queue management technique to control the delay when the queue size becomes large [16]. In

TABLE 2.2. Blue-RED algorithm.

Upon packet loss (or $q > L$) event:
if ((now – last_update) > freeze_time) then
$p_m = p_m + d1$
last-update = now
Upon link idle event:
if ((now - last_update) > freeze time) then
$p_m = p_m - d2$
last_update = now

order to control the frequency of changing the drop probability, Blue-RED uses extra parameters. The $freeze-time$ parameter determines the minimum time interval between two subsequent adjustments of p_m. Constant values of the $freeze–time$ cause global synchronization [28]. Thus, the value of the $freeze–time$ must be randomized. The parameters $d1$ and $d2$ determine the amount to increase and decrease the drop probability respectively. It is also recommended to set $d1$ significantly larger than $d2$ because the link utilization will be reduced whenever the queue management technique becomes aggressive or conservative.

2.10.2. *Other RED variants*

Many RED-based strategies were proposed with different parameter configuration. RED was originally designed to work with TCP/IP best-effort services. However, some RED variants were proposed and evaluated for differentiated services [11] such as multimedia UDP traffic [59] and ATM traffic control [63]. This section describes other RED-based strategies proposed in the literature. Some of these strategies have been evaluated in [10] [50] [52].

Dynamic RED (DRED): [3], tries to stabilize the queue size around a desired length (Q_{ref}) which is expected to stabilize link utilization at a preset level. The packet drop probability is adjusted according to the deviation of the queue length from Q_{ref}.

Stabilized RED (SRED): [56], drops packets according to the calculated number of flows and the current queue size. The number of flows

is calculated without maintaining a per-flow account. Buffer utilization in SRED is stabilized at a level which is independent from the load level.

Adaptive Virtual Queue (AVQ): [42], a modified packet model is used as a Virtual Queue (VQ), instead of the queue length to regulate buffer utilization. In case of VQ overflow, the size and the link capacity of the VQ are adjusted proportional to the estimated input rate and dropped packets.

2.11. Problems with AQM

Most of the Active Queue Management strategies use the average queue size to determine congestion. This causes some problems like the mismatch between macroscopic and microscopic behavior of queue length, insensitivity to the input traffic variations and some configuration problems.

2.11.1. *The mismatch between macroscopic and microscopic behavior of queue length*

The microscopic behavior of a router is the stable dynamics of the *average* queue size. These stable dynamics reflect the long term behavior of a router. In contrast, the short-term dynamics of the *actual* queue size are called the microscopic behavior of a router.

Some studies have shown varied dynamics between the average and actual queue sizes [10] [50] [56]. For instance, when a large number of bursts arrive at a RED gateway, the actual queue size is rapidly increased, resulting in buffer overflow. If the queue weight parameter is too small then the average queue size will be increased slowly, despite accumulating congestion. Sources will reduce their sending rate after a congestion signal is triggered due to packet drop at the gateway. After congestion problems have been rectified and the actual queue size is decreased, the average queue size will be high due to

previous peaks in the actual queue size. Therefore, packet dropping will be continued even after congestion problems have been rectified, which unfairly penalizes packets recieved after the congestion event.

2.11.2. *Insensitivity to the input traffic load variations*

The actual queue size is the best early indicator of congestion. Due to the use of the average queue size, the current variations in the traffic are not recognized by AQM strategies. This leads to unfair packet drops between connections. However, AQM strategies that tend to use the actual queue size to indicate congestion, instead of using the average queue size suffer from worst cases of unfair packet drops. Chapter 4 proposes a solution for this problem.

2.11.3. *Configuration problem*

Parameter configuration in AQM strategies is a difficult task, especially for RED. AQM modifications have been proposed to increase network performance which are evaluated using analytic modeling and simulation [15] [16] [44] [56] [19]. Unfortunately, these modifications work only for specific traffic conditions but not for realistic IP traffic [10] or a heterogeneous traffic environment [81]. Parameter configuration problems arise from the tradeoff between the design goals of AQM and the nature of the real IP traffic [64]. RED was proposed to accommodate excessive traffic load.

2.12. Congestion control approaches other than AQM

Active Queue Management is not the only approach that is employed to control congestion in current TCP networks. A few more approaches are also applied to solve this problem. Some of these approaches are architectural approaches, others are economic approaches.

Architectural approaches tend to enhance the source algorithm, network algorithm or both of them to provide better congestion control. These kinds

of approaches do not affect the design principles of TCP. Source algorithms use the implicit congestion notification without any assistance from the intermediate router. Network algorithms use the Explicit Congestion Notification (ECN). However, source and network algorithms can cooperate between each other to improve the network resource allocation as well as effective congestion control [31].

Using mathematical modeling and analysis, economic approaches have been proposed to provide better congestion control. The pricing-based approaches and the optimization approaches are subsets of the economic approaches. In the pricing-based approaches, packets or bandwidths are priced and used as congestion control devices for differentiation of services. The optimized-based approaches use game-theoretic techniques or mathematical programming to evaluate the behavior of users or networks. The following section describes one example for each set of approaches.

Dynamic Adaptive Window (DAW) [53] is an Architectural approach that provides a distributed algorithm to calculate end-to-end window size using analysis of closed queuing networks. The window size in DAW is adjusted according to packet-delay estimations instead of packet drop events. Congestion is determined by comparing the end-to-end delay to the range of acceptable delay.

Smart Market (SM) [47] is a pricing-based approach that allocates network resources based on packet prices. The prices of the packets are determined using the level of demand for link bandwidth. The network accepts packets with bid price that exceeds the instant cut-off amount. This amount is bounded by the marginal congestion cost imposed by the next additional packet. Unaccepted packets are returned back to the source or forwarded to another slow link.

Low [46] proposed an optimized based approach which considers links and sources as processors of distributed system. Data sending rates are selected by sources. Accordingly, network links determine the prices of the bandwidth to

organize the decisions that are made by sources. In addition, a Random Early Marking (REM) algorithm was proposed to use probabilistic packet marking using the explicit congestion notification bit in IP header [62] [18].

Economic approaches unlike architectural approaches require main changes in the design principles of TCP. Architectural approaches replace the packet drop event by ECN technique to indicate congestion. This thesis investigates network congestion control strategies that preserve the design principles of the TCP protocol. The thesis also investigates strategies that use the packet drop event to indicate congestion in intermediate router, such as TD and RED. Hence, architectural and economic approaches are out the scope of this study.

2.13. Summary

This Chapter presented the principles of TCP congestion control. The congestion control strategies are described from TD to the current state of the art AQM strategies. RED-based strategies suffer from three problems: the mismatch between macroscopic and microscopic behavior of queue length, parameter configuration and the insensitivity to the input traffic load variations. The first two problems are investigated in the subsequent chapters, and some solutions are proposed and evaluated.

CHAPTER 3

Queue Sectors RED (QSRED) Strategy

Traditional congestion control strategies use the actual queue size to control congestion in TCP gateways. In contrast, RED uses an Exponentially Weighted Moving Average (EWMA) of the queue size to control congestion. Some research suggests that there is a mismatch between the microscopic and macroscopic behavior in RED's queue management mechanism. This Chapter investigates this problem and proposes Queue Sectors RED (QSRED) strategy to enhance performance. QSRED is simulated against RED and ERED (Effective RED) by measuring: throughput, link utilization, packet loss and average delay using the NS2 simulator. The results suggest that Queue Sectors RED (QSRED) helps RED overcome the mismatch between microscopic and macroscopic behavior of queue length dynamics.

This Chapter is organized as follows: Section 3.1 presents a background about the use of the average and actual queue size parameters in TCP congestion control. Section 3.2 describes the relationship between the queue dynamics and buffer overflow. Section 3.3 describes the limitations of Effective RED (ERED). Section 3.4 presents some of the gaps in ERED's implementation. A novel strategy QSRED is proposed in Section 3.5. The network topology and simulation results are presented in Sections 3.6 and 3.7 respectively. Section 3.8 summarizes this Chapter.

3.1. Background

Abbasov and Korukoglu proposed ERED [1] and claimed that it provides greater throughput and packet send capability when compared with other RED variants. This research also states that other RED variants reduce their queue

size to zero when the buffer overflows, lowering network performance. In contrast, ERED only reduces the packet drops in case of buffer overflow. Abbasov and Korukoglu also state that other RED variants cause buffer overflows in the presence of bursty traffic. This Chapter investigates the algorithm and simulation scenarios of ERED and proposes enhancements.

Modern TCP routers implementing RED play a major role in queue management and congestion control. As described in Chapter 2, TCP routers dropped packets from the tail of the queue; hence called TD routers. TCP uses packet drops in intermediate routers as a signal of congestion. When a TD gateway overflows due to persistant congestion; all network sources reduce their sending rate simultaneously, reducing the link utilization. This problem is called global synchronization. ERED adds functionality to the classic RED algorithm to avoid this problem. This function is achieved by monitoring the actual queue size in the gateway.

Congestion control is subject to design strategies and algorithms that can dynamically control traffic sources when demand exceeds the available capacity. Enhancements to TD were proposed to avoid global synchronization. Therefore, new queue management approaches avoid the direct use of the actual queue length in estimating the drop level. Also, the new congestion control strategies aim to inform TCP senders of congestion before the buffer overflows. Newer congestion control mechanisms use the average queue size instead of the actual queue size to estimate the drop level. In addition, the drop probability is calculated to determine the drop rate which is increased/decreased whilst the average queue size traverses from low to high levels in the gateway buffer.

Queue management algorithms drop packets from the gateway buffer depending on a preset queue level (threshold). When the queue size exceeds this threshold, the gateway starts dropping packets to avoid congestion. Classic congestion control strategies monitor the actual queue size to handle congestion. Instead of the actual queue size check, the Random Early Detection (RED) maintains an Exponentially Weighted Moving Average (EWMA) to

monitor the queue dynamics. When the average queue size exceeds the threshold, RED drops packets randomly from the queue.

Research showed that there is a difference between the behaviours of the actual queue size and the average queue size [10] [56] [50]. It is possible for RED gateway to maintain a low average queue size whilst a peak in the actual queue size is exceeding the threshold and approaching the buffer limit. Hence, *the average queue size dynamics are referred by the macroscopic behavior of the queue.* Similarly, *the microscopic behavior of the queue denotes the dynamics of the actual queue size.* This Chapter discusses the mismatch between the microscopic and macroscopic behaviour of queue length dynamics in AQM strategies, particularly RED-based strategies.

3.2. The queue sizes mismatch and buffers overflow

Historically, congestion used to be controlled by network source mechanisms. When the intermediate router overflows and a packet is dropped, the source algorithm detects congestion and reduces the sending rate. As the field of congestion control evoloved, researchers noticed that the best place to control congestion is the gateway itself; because the gateway recognizes the dynamics of the actual queue size. Therefore, many gateway algorithms have been developed to detect the initial stages of congestion and avoid buffer overflow.

3.2.1. *Source algorithms and buffers overflow*

When router buffers become full, packets are dropped. TCP senders detect lost packets with duplicate acknowledgements. This means that *the source algorithm detects congestion after the intermediate router is overflowed and a packet is already dropped.*

Source algorithms are very useful for congestion control in TCP networks. However, there must be a strategy to manage the intermediate router queue

and inform the source of congestion implicitly or explicitly. Thus, TCP source algorithms works in conjunction with the intermediate router algorithms.

As mentioned in Chapter 2, traditional queue management strategies were simple. The gateway defines a drop level (threshold) which is normally the queue limit. Whenever the queue size exceeds this drop level, the gateway starts dropping packets in the order they arrive which is First-In First-Out queue management. One of the simplest and oldest implementations for queue management is the Tail Drop (TD); in which the packets are dropped from the tail of the queue. *A packet has to be dropped first due to buffer overflow, then TCP source detects congestion implicitly.*

The TD algorithm has three major problems. The first is the lock out problem. This occurs when a few nodes monopolize the whole network bandwidth [64]. The second problem is the full queue problem which occurs when a gateway continually sends full queue signals to sources for an extended period of time [64]. The third problem is the global synchronization when all TCP sources reduce their sending rate at the same time [29]; resulting in low throughput. Global synchronization is investigated in Chapter 5. Lock out and full queue problems are described further in Chapter 4.

3.2.2. *Gateway algorithms and the avoidance of buffer overflow*

As illustrated in section 3.2.1, gateways used to suffer from the buffer overflow problem. There was a need to detect the initial stages of congestion and inform the TCP source before buffer overflow. This approach is called the Active Queue Management (AQM). The Random Early Detection (RED) adopts the active queue management approach by maintaining an Exponentially Weighted Moving Average (EWMA) of the queue size on intermediate routers [29]. Instead of dropping packets based on the actual queue size, RED drops packets based on the average queue size avg. The drop rate of RED is increased/decreased using a drop probability p_a. Equations 2.3, 2.4 and

2.5 illustrate how RED calculates the average queue size avg, the initial drop probability p_b and the accumulative drop probability p_a respectively.

By maintaining two drop levels (the maximum and minimum thresholds), RED overcomes the drawbacks of TD. In order to control congestion efficiently, these two drop levels should work in conjunction with other parameters, such as the weight parameter w_q and the maximum drop probability max_p. However, parameter configuration in RED is difficult because it is dependent on the traffic type [51]. Parameter setting also depends on the number of active connections, buffer space limitations and the severity of congestion. For example, if the difference between the maximum and minimum thresholds is too small then the strategy operates similarly to the TD implementation. Conversely, if this difference is too big, then RED would mismatch the average and actual queue sizes; resulting in buffer overflow. It is the same for parameter w_q in Eq. 2.3. Small values for this parameter lead to a mismatch between the average and actual queue length dynamics.

RED also has a problem with the setting of the drop probability p_a which is a function of the parameter max_p. The parameter max_p is a constant in the traditional RED implementation. If the max_p parameter is assigned high value, then the drop probability p_a will be high; resulting in unfair packet drops. Therefore, some RED variants have proposed to use a dynamic max_p parameter [64] [15].

The estimated average queue size in RED does not reflect the mean of the actual queue size; rather it is a function of the weight parameter in Eq. 2.3. Smaller weight parameters exacerbate the mismatch between the average and actual queue sizes. When the mismatch happens, the actual queue size exceeds the available buffer size and a packet is dropped. For example, RED may not recognize a current peak in the actual queue size because it is working with the average queue size which is kept low. Immediately, the congestion is detected by the source algorithm due to packet drop using the TCP self-clocking technique. *In this case, network performance under RED*

may be similar to TD. Therefore, the mismatch between the microscopic and macroscopic behaviour of queue length dynamics causes three problems:

(1) The congestion avoidance technique used by RED will be switched back to congestion recovery by the source algorithm.

(2) Same as TD, RED will suffer from lock out, full queue and global synchronization.

(3) Higher drop rate is expected due to buffer overflow.

Figure 3.1 depicts the mismatch between the average and actual queue sizes. The average queue size, in RED gateways, is the macroscopic behavior of that queue because it reflects the long term dynamics of the queue length. Conversely, the microscopic behavior of queue reflects the short term dynamics or the actual queue size. Prior research has shown large differences between the average and the actual queue dynamics [10][50] [56].

Problems occur when traffic bursts arrive at an already congested gateway. If RED maintains a small weight parameter then the average queue size will be increased slowly. As a result, the gateway buffer overflows and packets are dropped. Congestion will be only detected by TCP sources. After congestion returns to the normal level, the average queue size will be increased. This can lead to unnecessary packet drops. These unnecessary drops cause the congestion window to be reduced after congestion has been alleviated and well be below the optimal level.

Effective RED (ERED) [1] strategy is a modification of RED. In this strategy, two drop levels $max`_{th}$ and $min`_{th}$ have been added to the normal RED algorithm. The strategy monitors both the average and actual queue sizes to avoid the mismatch between the average and actual queue sizes. This has the benefit of reduced buffer overflows and drop rates.

3.3. Effective RED (ERED) Strategy

Effective RED provides some enhancements to the classic RED. In small or empty queue stages, RED drops packets when the average queue size exceeds

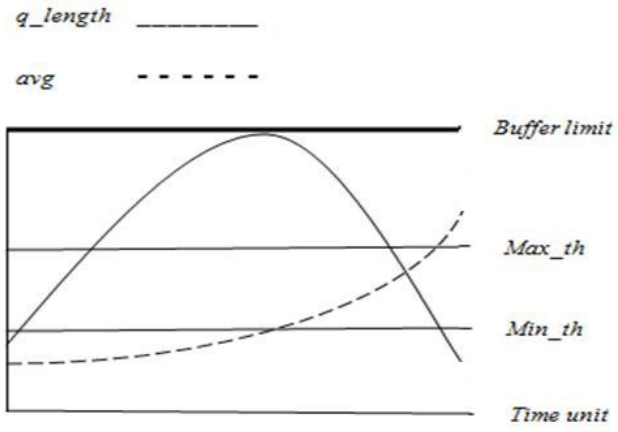


FIGURE 3.1. The mismatch between macroscopic and microscopic of queue length dynamics.

the minimum threshold. With heavier traffic loads, the actual queue size is increased quickly and approaches the queue limit which is an indicator that the buffer is overloading. The average queue size, in this case, is not large enough to make random drops.

ERED outperforms RED, allowing more aggressive packet dropping to quickly reduce the actual queue size before the buffer overflows. ERED [1] provides additional parameters to the traditional RED proposal. When the actual queue size is accumulating quickly and approaching the buffer limit associated with a low average queue size, ERED uses an aggressive packet drop technique to prevent buffer overflows.

3.3.1. *The RED implementation*

RED proposes that, if the average queue size is in between the minimum and maximum thresholds, then it indicates an initial stage of congestion and reacts by performing a random packet drop technique to avoid buffer overflow. When the average exceeds the maximum threshold; it means that the gateway suffers from severe congestion and the drop rate is not enough to regulate the send rate. In this case, all incoming packets are dropped. The question is: what will happen if the buffer becomes full but the average queue size is below the

maximum threshold?. This scenario is an example of the mismatch problem between the macroscopic and microscopic beviours of the average and actual queue sizes. ERED is proposed by Babek Abbasov and Serdar Korukoglu to avoid this problem. Section 3.3.3 describes the ERED's algorithm.

Abbasov and Korukoglu noticed that RED avoids overreacting to temporary traffic bursts and instead reacts to longer-term trends. They also noticed that, when RED thresholds are compared to the weighted average with a typical weighting factor (w_q of 1/512), it is possible that no forced drops will occur, even when the instantaneous queue length is quite large.

The random drop technique is triggered by RED when the average queue size exceeds the minimum threshold up to the maximum threshold. In contrast, forced drop technique for every arriving packet is performed by RED when the average exceeds the maximum threshold. The drop probability in the random drop area is calculated depending on other parameters which are: the initial drop probability p_b, max_p and avg.

When a new packet arrives and the average queue size is in between the minimum and maximum threshold, a random number $R \in [0, 1]$ from the uniform distribution on [0,1] is assigned to that packet. RED algorithm marks/drops the arriving packet if $R < p_a$. Abbasov and Korukoglu have another comment about this technique. The impact of a given average queue size is bounded by the value of min_{th} which depends on both max_p and max_{th}. All these parameters are preset by RED and the criteria of setting those parameters is not clearly determined. This means that one may find a value for min_{th} that results in good performance, but it may only be in combination with particular values of max_p and max_{th}. This can be generalized for all RED parameters. See section 2.9 for more details on RED.

3.3.2. *RED's shortcomings*

It has been shown that RED outperformed TD in many aspects. Abbasov and Korukoglu [1] discuss that there are some drawbacks in RED's performance. Using the NS2 simulator, they highlighted some weakness in RED's algorithm through analysis of simulation results for RED's parameters. They noticed the following malfunctions in RED performance:

- In the case of extreme congestion, RED still only marks a fraction (max_p) of the arriving packets and the gateway will fail to control the average queue size until it exceeds the maximum threshold. As a result, RED will drop every arriving packet until congestion is controlled.
- When the average queue length traverses from the minimum threshold to the maximum threshold, RED keeps marking/dropping the arriving packets with probability p_a, where p_a is a function of the average queue size. In this case, RED keeps dropping packets, even though the actual queue size is small or the buffer is empty.
- In case of congestion detection, connections reduce their sending rate immediately; resulting in quick decrease in the actual queue size whilst the average queue size is decreased slowly. Therefore, arriving packets will be unfairly dropped with higher drop probability due to the high average queue size. This problem is caused by false congestion notification.
- In case of heavy traffic loads, the queue will accumulate quickly and exceed the buffer limit whilst the average queue size is less than the minimum threshold, causing buffer overflow before the activation of the random drop technique.
- RED performance is reduced when the number of active connections increases.

- RED is very sensitive to traffic load variations and many queue oscillations are observed.
- RED performance is negatively affected by large packet sizes.
- RED behavior is determined by four parameters: min_{th}, max_{th}, max_p and w_q. Erroneous configuration for one of these parameters will reduce RED performance.

3.3.3. *ERED's algorithm*

ERED proposes some changes to the original RED algorithm to avoid the drawbacks in section 3.3.2 without changing the queue weight parameter w_q. ERED strategy adjusts the maximum threshold max_{th} and the minimum threshold min_{th} parameters. Eq. 3.7 and Eq. 3.8 illustrate how ERED sets these parameters.

$$max'_{th} = 2 * max_{th} \tag{3.7}$$

$$min'_{th} = \frac{max_{th} + min_{th}}{2} + min_{th} \tag{3.8}$$

If ($min'_{th} < avg < max'_{th}$) & ($qlen > min'_{th}$), ERED drops each arriving packet with probability p_a to match the current queue size with the average queue size. If ($avg < min'th$) & ($qlen > 1.75 max_{th}$) the strategy drops the arriving packets with probability p_b. This technique is used by ERED to avoid buffer overflow due to a peak in the actual queue size whilst the average queue size is less than the minimum threshold. Thus, ERED deffers from RED by applying the actual queue size check in addition to the value of the average queue size. If ($avg > K$) & ($qlen < T$) then $avg = N$, this is to avoid the misleading congestion notification. K , T and N are calculated by Eq. 3.9, Eq. 3.10 and Eq. 3.11 respectively. Table 3.1 illustrates the ERED algorithm.

Section 3.5 proposes the Queue Sectors RED (QSRED) which is a modification to overcome some of ERED's disadvantages.

$$K = \frac{max_{th} - min_{th}}{2} + min_{th} \tag{3.9}$$

$$T = \frac{max_{th} + min_{th}}{2} + max_{th} \tag{3.10}$$

$$N = \frac{2(max_{th} + min_{th})}{3} + min_{th} \tag{3.11}$$

3.3.4. *ERED evaluation*

ERED was evaluated [1] using the NS2 simulator. The network performance under ERED has been compared with the performance of BLUE, REM, SRED, LDC and FRED which are all AQM strategies. RED variants are normally tested using a bottleneck link topology, which is the same topology used by Abbasov and Korukoglu to evaluate ERED. Figure 3.2 illustrates this topology. In this topology, n TCP sources feed an intermediate router which is connected to a sink using a bottleneck link. In order to evaluate the impact of the traffic load variations, TCP sources produce different levels of traffic loads and hence different levels of congestion on the bottleneck link will be generated. The maximum size allowed for gateway queue is 50 packets.

Table 3.2 illustrates the parameters that are used for ERED evaluation. TCP sources used the exponential distribution for packet propagation with mean of 0.3s. Pareto distribution is used for file sizes with mean 10,000 and shape 1.5. Table 3.3 shows the performance evaluation for ERED and the other strategies included in the study.

Abbasov and Korukoglu have tested ERED under a per-flow congestion

TABLE 3.1. Detailed algorithm of ERED.

Initialization:
$avg = 0$
$count = -1$
for each packet arrival
calculate the new average queue size avg:
if the queue is nonempty
$avg = (1 - w_q)avg + w_q q$
else
$m = f(time - q_{time})$
$avg = (1 - w_q)^m avg$
if $(min'_{th} \leq avg < max'_{th})$ && $(q \geq min'_{th})$
increment count
calculate probability p_a:
$p_b = max_p(avg - min_{th})/(max_{th} - min_{th})$
$p_a = p_b/(1 - count.p_b)$
with probability p_a:
mark the arriving packet
$count = 0$
else if $(avg < min'_{th})$ && $(q > 1.75 * max_{th})$
calculate probability p_d:
$p_c = max_p$
$p_a = p_c/(1 - count.p_c)$
with probability p_d:
mark the arriving packet
$count = 0$
else if $(max'_{th} \leq avg\ )$
mark the arriving packet
$count = 0$
else $count = -1$
when queue becomes empty
$q_{time} = time$
for each packet departure
if $(avg > K)$ & $(q < T)$
$avg = N$

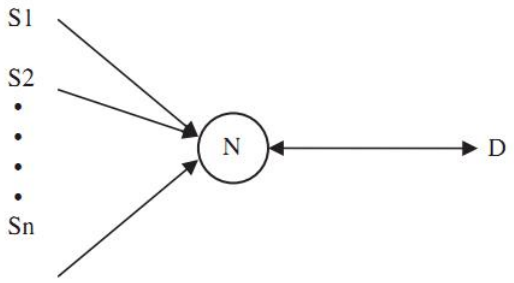


FIGURE 3.2. Simulation network topology to evaluate ERED's performance [1].

TABLE 3.2. Parameter configuration for ERED evaluation

Strategy	Parameter setting
RED	min_{th}=5, max_{th}=15, , w_q=0.002, max_p =1/150, $gentle = true$
SRED	$max_p = 0.15$, $p = 0.25$
FRED	$min_q = 4$
BLUE	$d1 = 0.00025$, $d2 = 0.0025$, $freeze - time = 100ms$
REM	ϕ=1.001, γ=0.001, w_q =0.002
LDC	D_{target}= 4s, w_q=0.002, w_r=0.998, n =3, R_{target}=0.95, τ=0.002 s, weights of 3/4 and 1/4 for the delay and load components respectively
ERED	min_{th}=5, max_{th}=15, w_q=0.002, max_p =1/150

TABLE 3.3. Performance evaluation for ERED and other RED variants

Strategy	Packets received	Packets sent	Packets dropped	Packet loss rate	Throughput
ERED	8370.143	8155.071	206.3571	0.0235	1.192354
SRED	8521.571	8149.929	324.7143	0.0352	1.192310
RED	8487.000	8018.6 43	463.4286	0.0612	1.170581
REM	8459.64 3	8144.214	304.4286	0.0356	1.190913
BLUE	8358.357	8055.500	295.2857	0.0362	1.176324
LDC	8448.357	8068.214	371.8571	0.0455	1.17936 8
FRED	8631.714	7280.500	1350.2860	0.1755	1.054913

control model. They ran their simulator for 50s and 100s and compared the loss rate, delay and jitters of ERED with other AQM strategies. Figure 5.16 illustrates the packet loss rate produced by 50s simulation time. In Fig. 3.4 they increased the simulation time to 100s and monitored the loss rate. In Fig. 3.11 and Fig. 3.12 they plot the network delay and jitter. Figures 3.5 to 3.10 depicts the actual queue length dynamics for all strategies included in their study.

Abbasov and Korukoglu claimed that ERED is the most conservative strategy. As the number of active flows increases ERED achieves the lowest loss rate, which is depicted in Fig. 3.4. The rest of the strategies were judged

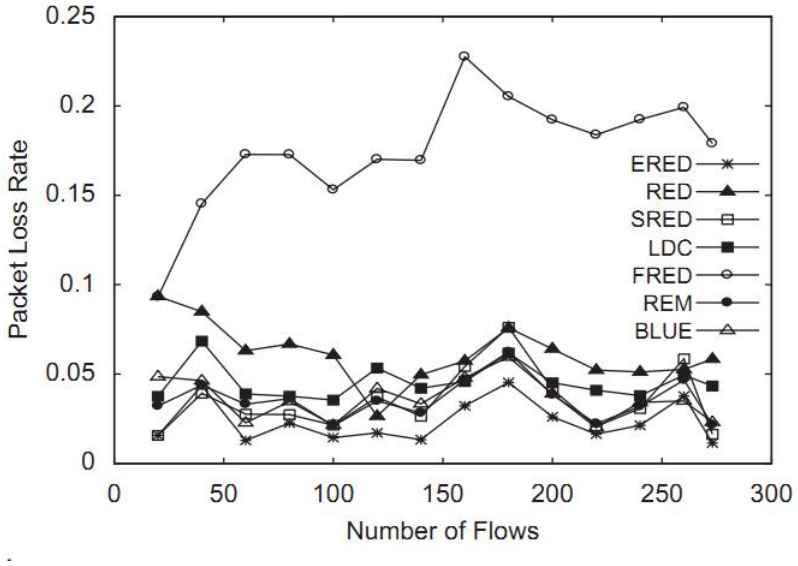


FIGURE 3.3. Packet loss rate versus number of flows for 50s simulation time [1].

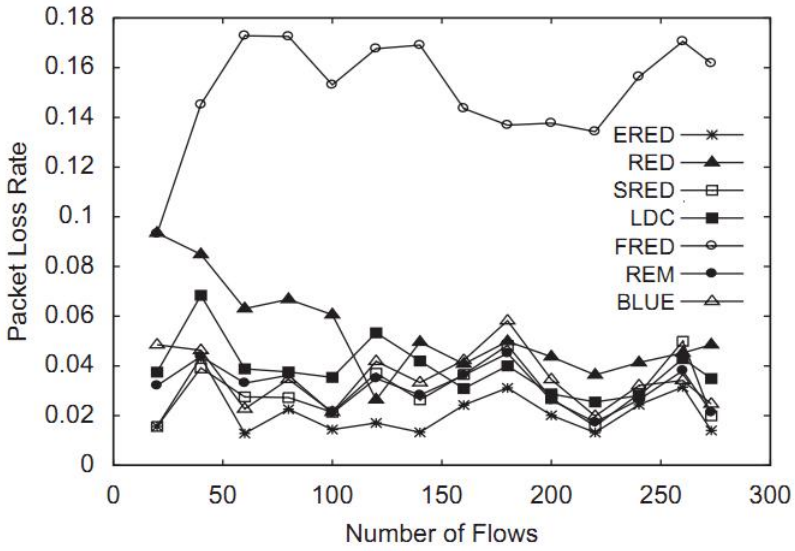


FIGURE 3.4. Packet loss rate versus number of flows for 100s simulation time [1].

against the loss rate and sorted from lowest to highest as: SRED, REM, BLUE, LDC, RED and FRED. This study [1] also shows that the refinements produced by ERED help RED reduce the packet lost rate.

Other performance parameters, such as ''total received packets'', ''total sent packets'', ''total number of dropped packets'', are demonstrated in Table 3.3. ERED dropped 206 packets out of the 8370 packets that arrived at the gateway during simulation time. The average throughput and average packet loss rate are 1.192354 and 0.23500 respectively. Thus, ERED outperformed the other strategies and produced superior network performance. Abbasov and Korukoglu also monitored the buffer queue size and claimed that ERED produces steady queue sizes, as illustrated in Fig. 3.5. The results in Fig. 3.5 suggest that strategies other than ERED produce more elastic queues. The packets are also dropped due to buffer overflow, as illustrated in Fig. 3.6 to Fig. 3.9.

In the same study, Abbasov and Korukoglu claimed that RED variants such as FRED often reduce the queue size to zero in case of heavy traffic loads to avoid buffer overflow. This technique could be successful to avoid buffer overflow as depicted in Fig. 3.10, has the negatve affect of reducing the link utilization and throughput and increasing packet loss. This is also illustrated in Table 3.3.

The authors in [1] proposed ERED as a compromise between aggressive AQM algorithms such as FRED and conservative strategies such as SRED. Congestion detection is more accurate in ERED by avoiding the misleading congestion notifications. In addition, ERED is more sensitive to the earlier stages of congestion. Section 3, describes ERED's implementation.

The delay and jitter under the ERED-AQM strategy are illustrated in Figures 3.11 and 3.12 respectively. As the number of flows increases, FRED shows the lowest delay and jitter values because of the high packet drop rate. RED and FRED were both considered as aggressive strategies by the authors of ERED [1]. The results suggest that RED and FRED produce low delay and

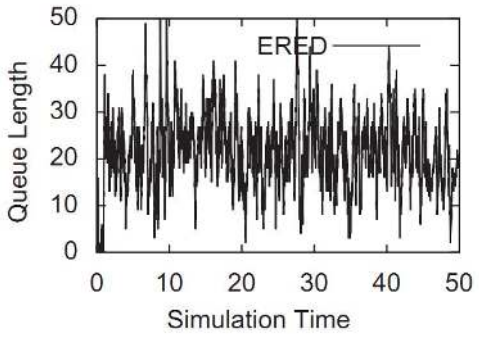


FIGURE 3.5. Queue length dynamics for ERED [1].

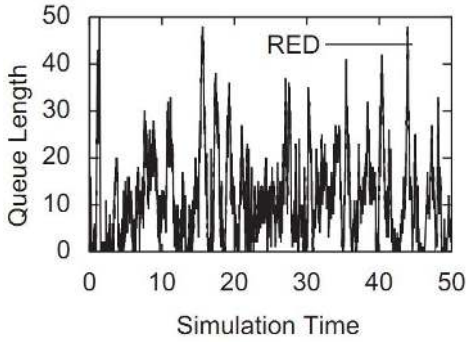


FIGURE 3.6. Queue length dynamics for RED [1].

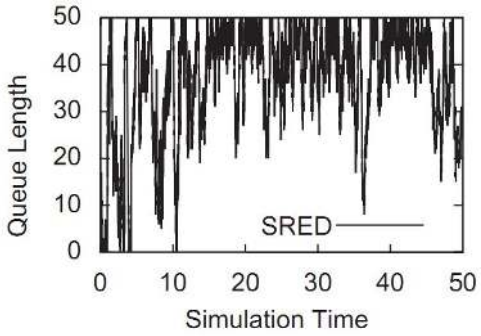


FIGURE 3.7. Queue length dynamics for SRED [1].

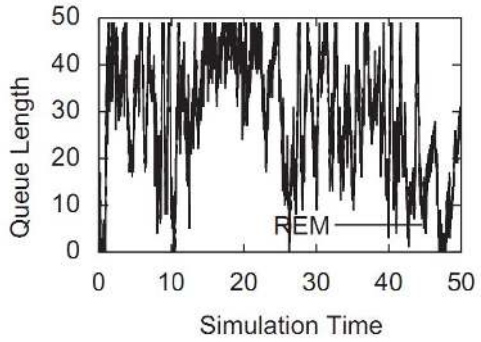


FIGURE 3.8. Queue length dynamics for REM [1].

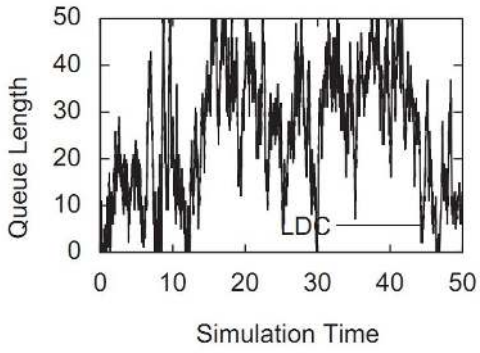


FIGURE 3.9. Queue length dynamics for LCD [1].

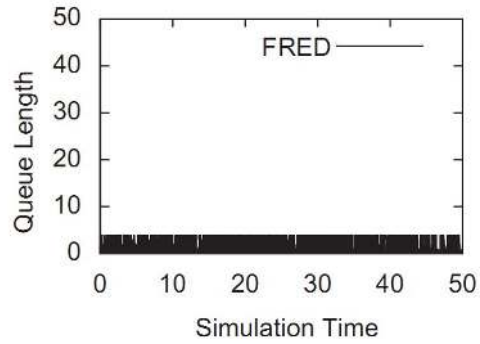


FIGURE 3.10. Queue length dynamics for FRED [1].

jitter values.

3.4. Gaps in ERED's evaluation

The performance of ERED was evaluated against the loss rate. Simulations have shown that ERED reduces the loss rate. This section investigates the parameter configuration that was used to evaluate ERED, particularly the maximum drop probability max_p and the actual queue size check.

3.4.1. *Parameter configuration*

The maximum drop probability max_p is one of the main parameters that affect the overall performance of RED variants. Table 3.2 illustrates the parameter configuration used in [1]. The following sections consider the accuracy of the max_p parameter. Also the configurations of the maximum threshold, minimum threshold and the drop probability are investigated.

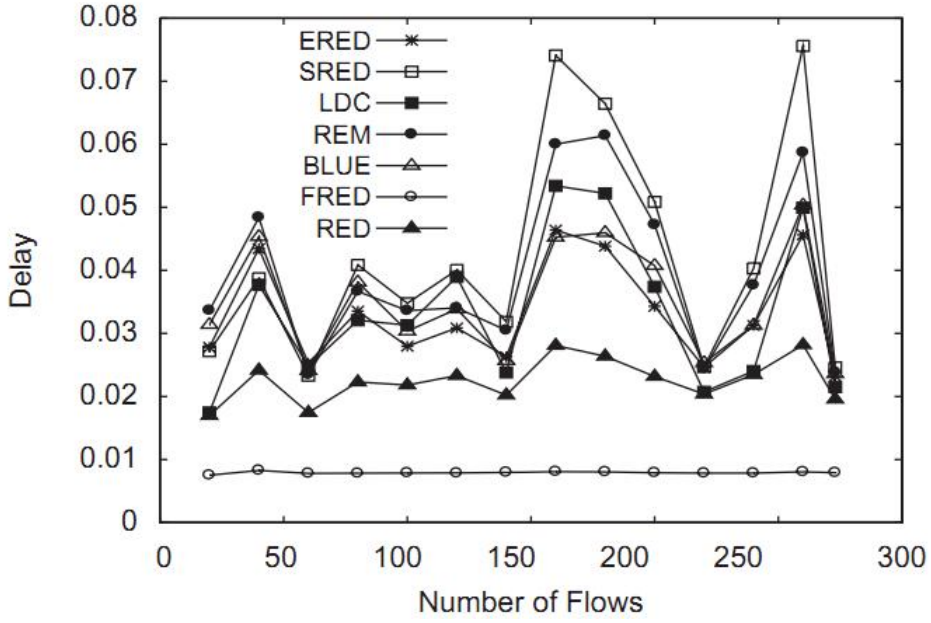


FIGURE 3.11. Compression between ERED's delay and other AQM algorithms [1].

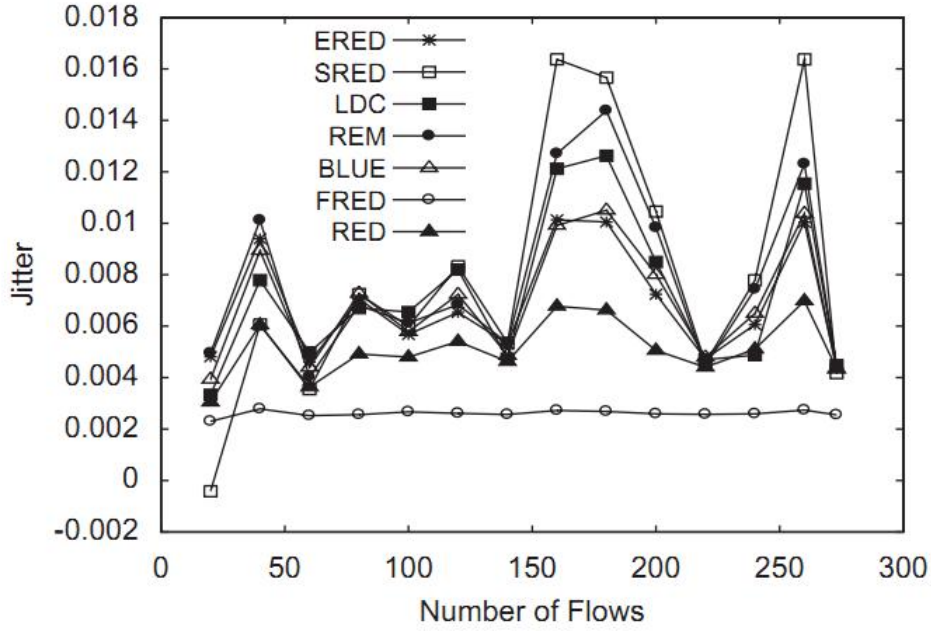


FIGURE 3.12. Compression between ERED's jitter and other AQM algorithms [1].

3.4.1.1. *Maximum drop probability max_p:* The role of max_p is active when the average queue size is in between the minimum threshold and maximum threshold. When the average queue size is halfway between min_{th} and max_{th}, the gateways drops roughly one out of $1/max_p$ packets. For example, if the maximum drop probability is preset to 0.1, the gateway drops on average, one out of 10 of the arriving packets. When the average queue size approaches the maximum threshold, RED will mark or drop nearly one fifth of the arriving packets. Thus, the robustness of RED lies in the slow changes in the drop probability depending on the average queue size changes. In case of heavy congestion, the average queue size will be preemptively increased. When the average queue size exceeds the maximum threshold, the drop probability will be set to one. In other words, the gateway marks/drops every arriving packet.

Floyd and Jacobsen in [29] recommend that setting max_p greater than 0.1 will increase the drop rate and make RED too aggressive against bursty traffic. Conversely, max_p values less than 0.01 will unfairly reduce the drop probability. Abbasov and Korukoglu set max_p to 1/150 in Table 3.2 which is nearly equal to 0.0067. This was compared with SRED where the max_p was set to 0.15. The configuration for this parameter is not clear for the rest of the strategies in the same table, except for RED which was set to 1/150. However, the parameter *gentle* is set to *true* for RED and this option multiplies max_p when the average queue size exceeds a predefined threshold. This explains the low drop rate for ERED comparing with the other strategies included in the study.

3.4.1.2. *Maximum and minimum thresholds (max_{th} and min_{th}).* RED works effectively when the difference between the maximum threshold and minimum threshold is larger than the typical increase in the calculated average queue size in one round-trip time [29]. The most efficient values for the maximum and minimum thresholds are estimated relating to the desired average queue size. The authors in [1] set the maximum threshold to twice the minimum threshold which is just a rule-of-thumb recommended for normal traffic in [29].

For bursty traffic, it is suggested to set a large minimum threshold to increase link utilization. For traffic in ERED's simulations, the minimum threshold of five packets would result in low link utilization. The min_{th} parameter was not investigated in [1].

The maximum threshold parameter should be set depending on the maximum average delay permitted by the gateway. In addition there should be enough space between the maximum threshold and minimum threshold in order to absorb the short-lived bursty traffic and transient congestion. Hence, it would be better to set the maximum threshold, depending on the gateway buffer space, for heavy traffic loads simulated in [1].

3.4.1.3. *Actual and initial drop probabilities (p_a and p_b).* ERED applies a technique to check the mismatch between the behaviours of the actual queue size and the average queue size. When the average queue size is less than the min'_{th} and the actual queue size is nearly twice the maximum threshold ($1.75max_{th}$), then ERED drops the arriving packets with probability p_b instead of probability p_a. In this case, ERED gateway considers the peak in the actual queue size as a congestion indication and replaces the actual drop probability with the initial drop probability to avoid buffer overflow. This scenario has more than one drawback:

First, the maximum threshold used to evaluate ERED in Table 3.2 is 15 packets. Abbasov and Korukoglu used Eq. 3.8 to calculate the parameter min'_{th}, also 15 packets. RED drop technique is disabled when the average queue size is less than the actual minimum threshold (min_{th}). The actual queue size check is applied by ERED when the average queue size is in between the normal minimum and maximum thresholds (min_{th} and max_{th}). In this situation, ERED will work same as the classic RED strategy.

Second, it has been shown in section 3.4.1.1 that the parameter max_p used in [1] was too small. The study also shows that the lower maximum drop probability is the lower drop rate for the RED variant. This explains the low loss rate achieved by ERED in Table 3.3.

Third, when the average queue size traverses between the minimum and maximum thresholds, RED uses Eq. 2.4 to calculate the initial drop probability p_b. The parameter *count* represents the number of dropped packets since the last dropped packet. The *count* parameter has to be included in the calculation of the final drop probability p_a, in Eq. 2.5. By doing so, RED fairly distribute the intervals between dropped packets to avoid global synchronization [29]. It is recommended in [29] that the number of arriving packets between successive marked packets should be a uniform random variable. Hence, the use of the initial drop probability p_b instead of the accumulative drop probability p_a in ERED will cause global synchronization. Also, the value of p_b is always less than the value of p_a. Therefore, ERED is always less aggressive than other RED variants and drops fewer packets, with the higher risk of global synchronization.

3.4.2. *Actual queue size check*

RED divides the gateway buffer into three sectors and calculates the average queue size. Figure 3.13 illustrates RED's sectors. Sector one covers the area between the empty queue and the minimum threshold. If the average queue size is in this area, RED does not apply any action. The second sector covers the area between the minimum threshold and maximum threshold. If the average is in this sector then RED calculates the initial and accumulative drop probabilities and starts dropping packets using the accumulative drop probability. If the average queue size exceeds the maximum threshold then all arriving packets will be dropped. In contrast, ERED divides the gateway buffer into seven sectors which are illustrated in Fig. 3.14. ERED monitors both the average and actual queue sizes in three positions. Following is a comparison between the functions of RED and ERED in the permutations of actual and average queue lengths:

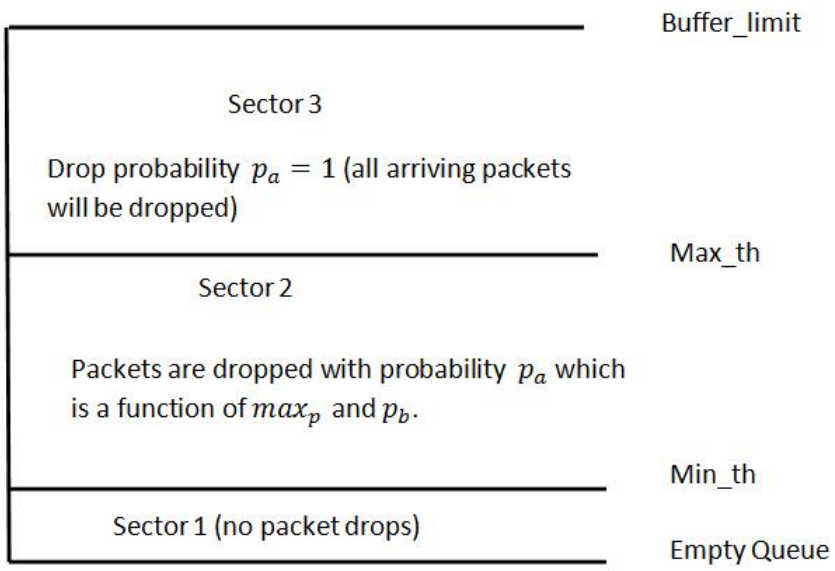


FIGURE 3.13. RED gateway sectors

3.4.2.1. *Position one.* If the average queue size is greater than min'_{th} but not exceeding max'_{th} and the actual queue size is greater than min'_{th}, then ERED determines congestion is moderate. In response, ERED drops the arriving packets with probability p_a.

Suppose that the average queue size is already in sector 6 and the actual queue size is in sector 7, then position one is satisfied. The average queue size reflects the long term congestion history. As long as the average queue size reaches this sector, then the gateway has been already congested for a long period of time. In this case, the gateway suffers from persistant congestion and buffer overflow is unquestionable. ERED uses the last actual drop probability p_a to drop packets from the gateway. In contrast, RED will apply an earlier proactive approach. When the average exceeds the maximum threshold, RED will drop every arriving packet until the queue is reduced below the maximum threshold.

3.4.2.2. *Position two.* ERED also applies a misleading congestion check. If the average queue size is greater than K, which is calculated in Eq. 3.9, and the actual queue size is less than T, then ERED mistakenly indicates congestion. In response, ERED stabilizes the average queue size at N which is calculated

using Eq. 3.11. Stabilizing the average queue size at a particular level, grants a uniform distribution for packet dropping which reduces the risk of global synchronization. This stabilization is perfect when the average queue size is roughly halfway between the minimum and maximum threshold with adequate setting for the maximum drop probability [29]. ERED in this position will maintain high average queue size. For example, the minimum threshold and maximum thresholds used to evaluate ERED in [1] were 5 and 15 respectively. Thus, ERED will assign the value 18 for N using Eq. 3.11. In this case, the average queue size will be greater than the maximum threshold and every arriving packet will be dropped. Therefore, ERED would not outperform RED in this position unless inequality 3.12 is satisfied.

$$min_{th} \leq N \leq max_{th} \tag{3.12}$$

3.4.2.3. *Position three.* The final check for ERED is when the average queue size is less than K (sectors 1 and 2) and the actual queue size is greater than $1.75max_{th}$ (approaching sector 7). In this case, ERED drops packets with probability p_b. The average queue size is very low whilst a peak in the actual queue size is approaching the queue limit. Sector one should not be investigated in this position because RED variants do not drop packets when the average queue size is less than the minimum threshold. Sector two belongs to the area between the minimum and maximum thresholds. In this area, RED normally drops packets with the accumulative drop probability p_a which is always higher than the initial drop probability p_b that is used by ERED. Apart from the global synchronization that can be incurred by using the initial drop probability p_b the drop rate of ERED will be less than the drop rate of RED. This was illustrated in section 3.4.1.3.

There are three possible scenarios for ERED's performance depending on the nature of the peak in the actual queue size:

(1) Short lived bursty traffic: if the peak is generated by short lived bursty traffic, then RED will be more aggressive than ERED and unnecessarily drop packets.

(2) Transient congestion: if the peak in the actual queue size is generated by transient congestion, then there is no need to increase the drop rate. Therefore, ERED with low drop probability p_b will outperform RED with high drop probability p_a.

(3) Persistent congestion: the average queue size reflects the long-term queue dynamics. The gateway will be misled in detecting the long-term congestion due to the previous history of the queue. This means that the gateway has not suffered from any serious congestion problems for long period of time and the queue will continue in steady state. When the persistent congestion starts with a peak in the actual queue length, the gateway would not recognize the congestion. To solve this mismatch between the average queue size and the actual queue size, the drop rate must be increased quickly before the buffer overflows. There are two options to increase the drop rate.

The first option is to increase the drop probability. In this case, the drop rate of RED will be higher than the drop rate of ERED. Hence, RED will be better than ERED in controlling the persistent congestion. The parameter max_p can play a main role in increasing and decreasing the drop probability. If the gateway maintains adjustable max_p parameter, then the drop rate can be increased and decreased depending on the state of congestion. Abbasov and Korukoglun used small max_P value to evaluate ERED which makes the problem of the persistent congestion worse. Section 3.5 proposes new dynamic maximum drop probability (max_P) to solve this problem.

The second option is to increase the average queue size itself, because the drop rate in upper sectors is higher than the drop rate in lower sectors. Some RED variants maintain preset parameters to increase and decrease the average

queue size. Chapter 5 proposes a new technique to increase the sensitivity of the average queue size to changes in the actual queue size. This is achieved by controlling the the weight parameter (w_q) in Eq. 2.3.

3.4.3. *Queue Sectors*

There are some drawbacks in ERED's drop levels (sectors). Table 3.4 illustrates a sample of the possible configuration for ERED drop levels. The problem with these thresholds that they mainly depend on the actual maximum threshold (max_{th}) and minimum threshold (min_{th}). It has been shown that the setting of these parameters is problematic [29], complicating the design of ERED. Suppose, for example, that the buffer limit in ERED gateway is 60 packets. It is easy then to find some scenarios in Table 3.4 where some of the drop levels exceed the buffer limit. In this case, the gateway will not apply the ERED algorithm for congestion control; rather it goes back to the normal RED algorithm.

The job of the thresholds is very critical, because they determine the time and rate of packets drop. Thus, any erroneous configuration for the drop level will affect the overall performance of the network. Regardless of the even distribution for ERED sectors in Fig. 3.14, the real distribution could be inconsistent. The drop levels can be intersected in the current configuration of ERED. For example, in scenario 1, Table 3.4, the parameters max_{th} and min'_{th} intersect which will disable the function of one of the thresholds. Section 3.5 presents more consistent distribution for ERED sectors.

3.5. QSRED Algorithm

The novel approach in QSRED is to divide the gateway buffer into six equal sectors. The drop probability and max_p parameters are to be adjusted when the actual and average queue sizes traverse between these sectors. This section shows how this approach improves network performance and stability.

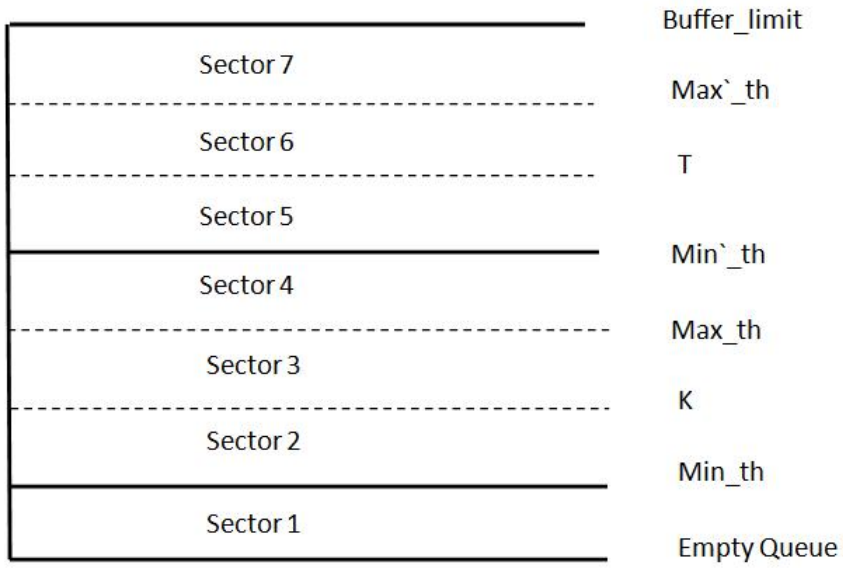


FIGURE 3.14. ERED sectors

TABLE 3.4. Sample of ERED's sectors

Scenario	min_{th}	K	max_{th}	$min`_{th}$	T	$max`_{th}$	N
1	5	10	15	15	25	30	18.33333
2	10	15	20	25	35	40	30.00000
3	17	21	25	38	46	50	45.00000
4	25	27.5	30	52.5	57.5	60	61.66666
5	30	35	40	65	75	80	76.66666
6	50	75	100	125	175	200	150.00000
7	60	100	140	160	240	280	193.33333

ERED added adjustable max_{th} and min_{th} parameters; which matches the microscopic and macroscopic behaviours of the queue in two situations. However, ERED still has some drawbacks:

(1) ERED uses the normal parameters of the original RED. It is clear from Eq. 3.7 and Eq.3.8 that max'_{th} and min'_{th} are functions of the normal max_{th} and min_{th}. Those parameters have the same configuration problem as RED [51].

(2) The strategy would be more powerful if the max_{th} and min_{th} are adjustable during the simulation run time in response to the traffic load dynamics.

(3) ERED proposed to keep low loss rate values but does not make any

calculation for the other three performance factors: throughput, link utilization and delay.

(4) In Eq. 3.13, if the inequality is satisfied then ERED uses the immediate marking probability, which is a function of avg and max_p. The value of max_p in this case, is minimally smaller, resulting in a lower dropping rate. Hence, the risk of buffer overflows is high. In such a serious situation, congestion control strategy should increase the dropping probability to a value approaching 1. This will allow the buffer to be drained before it becomes overloaded.

(5) Equally, if the queue length is greater than the drop level ($1.75max_{th}$) then it drops the arriving packets with probability p_b. It is also clear from the previous literature that there are no suggestions about the optimal setting of this drop level.

Accordingly, we propose the following parameter configuration to enhance ERED's functionality:

Firstly, one of the best ways to avoid using max'_{th} and min'_{th} as functions of the actual max_{th} and min_{th} is to choose some parameters relating to the buffer size. There is no adequate way to set the max_{th} and min_{th} of a queue. However, if the gateway operates ERED, it is advantageous to use smaller max_{th} and min_{th} parameters to avoid exceeding the buffer size when calculating max'_{th} and min'_{th} parameters.

Secondly, there is an option to divide the buffer into 6 equal sectors. Then, a reasonable value for the max'_{th} could be 4/6 buffer size, keeping the remaining 2/6 buffer size for short lived bursty traffic. The parameter max_{th} can be equal to 3/6 (half of the buffer size). The parameters min'_{th} and min_{th} can be 2/6 and 1/6 buffer size respectively.

Finally, to avoid the drawbacks (3) and (5) of ERED, it is advisable to increase the drop probability to help the gateway drop the overloading packets. If the inequalities in equations 3.13 and 3.14 are satisfied then it is suggested to

TABLE 3.5. QSRED algorithm

** for every arriving packet
if (q_size >= Sec.5 and avg <= Sec.2)
Then
$max_p = 2 * max_p$
Else
Go To: Traditional RED Implementation

set max_p to $2 * max_p$. In addition, the inequality in Eq. 3.13 is replaced by the inequality in Eq. 3.15 to execute the actual queue size check. Consequently, the parameter ($1.75max_{th}$) in inequality 3.14 of ERED is replaced by the parameter $(5/6 * buffer - size)$. Also, the normal drop probability p_a is used instead of using the current small dropping probability p_b. This has the effect of speeding up the queue drain process in case of congestion. Table 3.5 shows the algorithm of QSRED.

$$(qlen > 1.75max_{th}) \tag{3.13}$$

$$(avg < min'_{th}) \tag{3.14}$$

$$(ERED\_Q > 5/6 * buffer - size) \tag{3.15}$$

Consider, as an example, a buffer with size 90 packets. Suppose that, $avg = 25$, $qlen = 80$ and the initial $max_p = 0.01$. Then, both of the inequalities 3.13 and 3.14 are satisfied. ERED's parameter configuration will be as in Table 3.2. In this situation, the queue is rapidly accumulating and the buffer is about to be overloaded while ERED keeps a small max_p value. This value is not big enough to regulate the queue size. For better queue management performance, the drop probability parameter is increased by a value that allows the queue to return to the normal level (see p_b, for the QSRED).

TABLE 3.6. Parameter configuration for ERED and QSRED

	ERED	QSRED
min_{th}	20	15
max_{th}	40	45
min'_{th}	50	30
max'_{th}	80	60
p_b	$0.25max_p$	$0.67max_p$
p_a	0.0025	0.00745

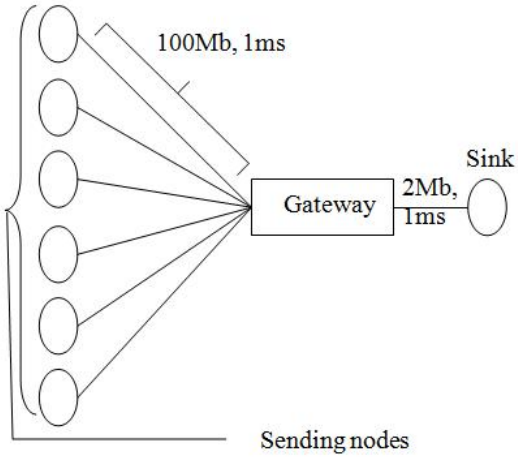


FIGURE 3.15. The simulator network topology.

QSRED monitors the actual and the average queue size values. If the actual queue size is below sector five and the average queue size is below sector two, the maximum drop probability parameter max_p is duplicated and the accumulative drop probability p_a is used instead of using the current drop probability p_b to drop the arriving packets. In this scenario, p_a is the best drop probability because its value is higher than p_b. This helps to shrink the queue quickly.

3.6. Network Topology

Fig. 3.15 illustrates the network topology used to test QSRED with four network performance parameters. In this topology, six sources send packets with a maximum window size equal to 2000 Bytes. The TCP version used here is TCP-Reno. A sink immediately sends an acknowledgment packet when it receives a data packet. Arrivals of sessions follow a Poisson distribution.

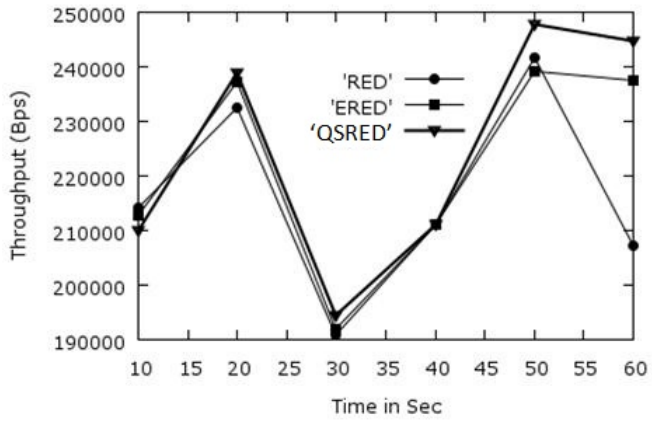


FIGURE 3.16. Network throughput.

A connection between each node and the gateway has 1ms delay time. The bottleneck link between the gateway and the sink has a 1ms delay time for delivering the packet to the sink. Exponential distribution is used for the start time of packets transfers.

The following section describes the simulation results over RED, ERED and QSRED with respect to four network performance parameters.

3.7. Simulation Results

Figures 3.16 to 3.19, depict four network performance parameters for RED, ERED and QSRED strategies of congestion handling. These four parameters are the network throughput, link utilization, packets loss rate and average delay time respectively. In addition, Fig. 3.20 shows network jitter. The simulation results are performed using the NS2 simulator [37]. The network topology used in this simulator is depicted in Fig. 3.15, which is roughly the same topology used by Abbasov and Korukoglu to evaluate ERED in [1].

Overall, the figures show that QSRED slightly outperforms RED and ERED. QSRED has better throughput, which is indicative of higher performance over the other two strategies. Additionally, QSRED shows a higher level of stability over RED and ERED. QSRED also reduces the packets loss rate whilst RED and ERED curves depict higher loss rates.

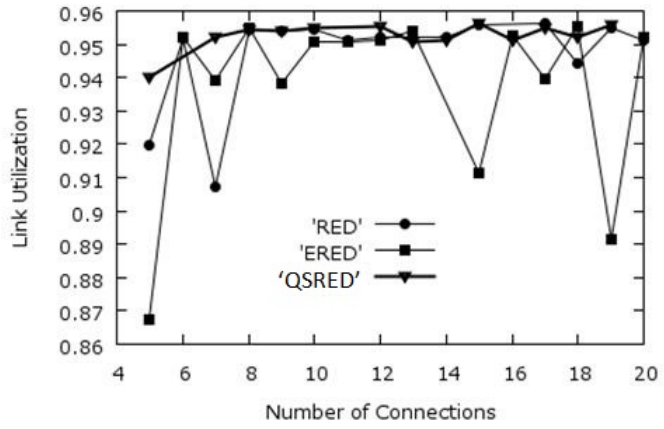


FIGURE 3.17. Network link utilization.

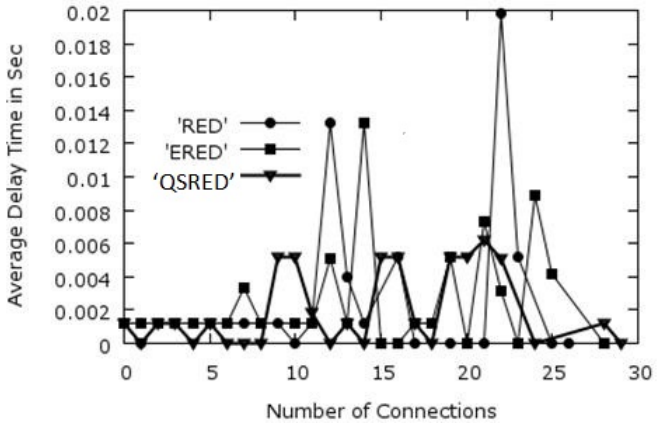


FIGURE 3.18. Average network delay.

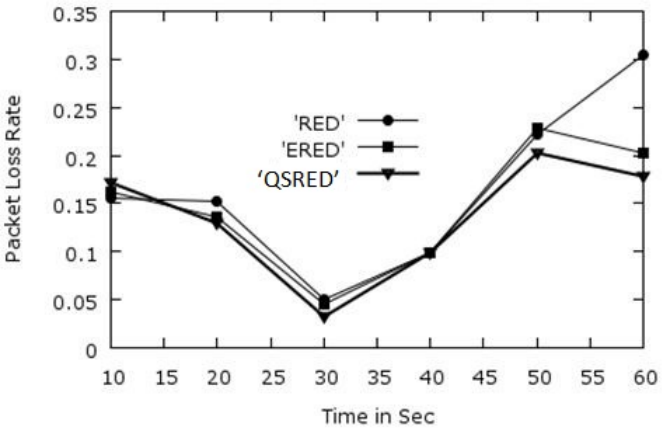


FIGURE 3.19. Packet loss rate.

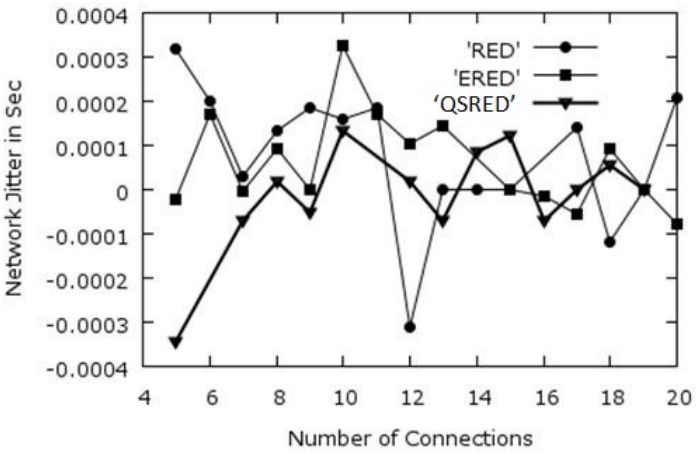


FIGURE 3.20. Network jitter.

One of the main goals of the RED proposal is to overcome the global synchronization problem. In the presence of bursty traffic, like the one used in this simulator, global synchronization is still possible. It is clear from the throughput figure (Fig. 3.16) that the network suffered from global synchronization problem at time 20s. All sources reduced their sending rates at the same time for RED, ERED and QSRED. In this case, the gateway suffered from congestion. The gateway starts to recover at time 30s. Again, the gateway suffered from serious congestion level at time 50s which is the best time to examine the performance of each strategy.

All strategies started propagation at the normal rate and the network was steady until time 20s when the first congestion was occurred. The loss rate, shown in Fig. 3.19, illustrates which strategy predicted congestion earlier. The period of time that must be investigated, for this purpose, is the time from 10s to 20s. During this period, the drop rate was steady for RED's figure which is followed by a sudden reduction in drop rate at time 20s. This indicates that the average queue size was moving from the minimum threshold to the maximum threshold in this period. All strategies intersected at time 15s which means that ERED and QSRED have already detected congestion at time 15s and reduced their sending rate while RED continues dropping at normal drop rate. At time 20s the average queue size hits the maximum

threshold. Hence, all strategies have to reduce the sending rate. It is clear that QSRED outperformed both RED and ERED in this stage and achieved higher throughput with fewer packet losses.

The gateway started to recover between the 30 and 50 seconds interval. The congestion at this time is less aggressive than the previous one. However, RED has dramatically increased the drop rate and the throughput reduced as a result. This happened because of the high average queue size from the previous history of the gateway. ERED and QSRED maintain low queue size with conservative throughput.

With respect to the average delay time parameter, there were a few intersections in the curves of the three strategies. Despite these intersections, QSRED lowerd overall delay during the simulation. This has been reflected in Fig. 3.20. QSRED appears the most stable whereas RED and ERED results show greater variance.

Figure 3.16 shows the throughput parameter for RED, ERED and QSRED in bytes per second (Bps). The QSRED strategy has shown the best throughput values. It is clear that RED achieved the lowest throughput values, particularly at the end of the simulation when throughput reached 207,000 Bps. It was observed that ERED throughput is higher than RED but it is still smaller than QSRED.

Figure 3.17 plots the link utilization parameter. It demonstrates that QSRED link utilization is almost stable with the number of connections. In contrast, ERED and RED link utilization fluctuate dramatically. It is also noticeable that QSRED shows the highest link utilization with variant number of connections.

Another important performance parameter, demonstrated in Fig. 3.19, is the packet loss rate. QSRED reduced the packet loss rate to 4% of the total number of packets propagated at time 30s. For RED, the loss rate was very high and reached 30% of the total propagated packets at the end of the simulation.

Figure 3.18 illustrates the average network delay. Despite the oscillations with increasing number of connections, the results suggest that QSRED adds the lowest amount of delay. This suggests that QSRED maintains shorter average queue sizes. The figure shows that the maximum delay of RED exceeds 0.019s. For ERED this value exceeds 0.013s but with QSRED this value did not exceed 0.007s. Furthermore, QSRED delay time has been kept below 0.001s for the majority of simulation time.

Figure 3.20 shows the amount of jitter introduced by the AQM strategies. Although the results appear similar, QSRED expressed the best network jitter results. The results, show that the minimum value for QSRED approaches -0.35ms and the maximum value is close to 0.13ms. Conversely, the values for ERED and RED ranged between -0.7ms to 0.33ms and -0.31ms to 0.33ms respectively. A qualitative analysis shows that the QSRED curve is the least erratic.

3.8. Summary

This Chapter introduced an efficient way to overcome the mismatch between the microscopic and macroscopic behaviors of queue length dynamics in RED gateways. The Chapter introduced a new algorithm, QSRED, that enhanced the implementation of the original RED strategy. RED has difficulties detecting this mismatch. This can result in RED continuing to drop packets after congestion event has subsided. This work provides a new technique that helps the gateway manage network congestion and increase TCP performance.

The new algorithm, Queue Sectors RED (QSRED), divides the buffer into six equal sectors. These sectors represent new dropping levels added to the original RED implementation. It uses the actual and the average queue size parameters to help RED absorb short lived bursty traffic and control TCP congestion efficiently. Since RED uses probabilistic packet dropping, QSRED dynamically adjusts the max_p value of RED to maintain network stability and smooth traffic. This chapter compared QSRED with ERED and RED

strategies in NS2. The results show that QSRED offers higher throughput and link utilization with lower packet loss and delays.

CHAPTER 4

A Third Drop Level For RED

A new strategy called Risk Threshold RED (RTRED) is proposed in this Chapter. In addition to the maximum threshold and minimum threshold in RED, RTRED adds an extra threshold (the risk threshold) and applies average and actual queue size check. Instead of swapping between the accumulative drop probability p_a and the initial drop probability p_b to control the drop rate in ERED, RTRED adjusts the drop rate dynamically by maintaining adjustable maximum drop probability max_p. RTRED algorithm is expected to avoid the unnecessary packet drops in RED gateways. Packet drops due to lock out and full queue problems are further investigated in this chapter.

The key idea of congestion control is to determine an appropriate congestion level to start packet dropping. Unfortunately, there is no strategy that provides perfect balance. This Chapter shows how RTRED solves some of the imbalances in previous strategies. The Chapter also shows how RTRED better balances packet loss, average delay and queue space utilization. The new dropping level is the risk threshold which works with the actual and average queue sizes to detect the immediate congestion in gateways. RTRED aims to react to congestion neither too early, to avoid unfair packet losses, nor too late to avoid packet dropping from time-outs. This novel strategy is compared with RED and ARED strategies for TCP congestion handling using a NS-2 simulation script.

Chapter 4 is organized as follows: section 4.2 describes the lock out and full queue problems. Sections 4.3 and 5.4 describes ARED and FRED strategies respectively. Section 4.4 proposes the Risk Threshold RED (RTRED). The network topology and simulation results are presented in sections 4.5 and 4.6 respectively. The chapter is summarized in section 4.7.

4.1. Background

Chapter 2 describes how current high speed network gateways are likely to be congested due to the increased demand for bandwidth. Early TCP congestion control strategies attempted to manage congestion by manipulating the congestion window size *cwnd*; which is a parameter that regulates the sending rate [64].

The goals of any congestion control strategy are: (I) Fair resource allocation (II) Reasonable queuing delay (III) minimal packet loss and (IV) Low resource consumption. These four goals are conflicting. For example, if a strategy is designed to reduce the packet loss rate at the gateway, higher queue sizes will be produced. Higher queues on routers increase end-to-end delays. This Chapter, proposes the Risk Threshold RED (RTRED) strategy to better balance these conflicting goals.

Traditional congestion control strategies, such as TD, used the actual queue size to check congestion in intermediate routers. If the actual queue size exceeds a preset threshold, then all incoming packets are dropped until the congestion has been alleviated. This technique causes many problems, such as global synchronization, lock out and full queue. These problems increase the drop rate to an unacceptable level and packets will be dropped unfairly in the gateway.

To solve these problems, congestion control strategies, such as RED, replace the actual queue size check with the average queue size check. Chapter 3 addressed the drawbacks associated with the implementation of Random Early Detection (RED) for congestion handling in TCP networks. It has been shown that there is a mismatch between the behaviours of the average and actual queue sizes; resulting in serious problems in the gateway. Even with the average queue size check, it has been shown in [1] that RED gateways still suffer from lock out and full queues due to the mismatch between the average and actual queue sizes. Abbasov and Korukoglu proposed Effective RED (ERED)

[1] strategy to reduce packet loss caused by this mismatch. The gateway is divided into seven sectors and both of the average and actual queue sizes are monitored. If the mismatch is caused by the average queue size then, ERED adjusts the average queue size to match the actual queue size. Otherwise, the mismatch is caused by the actual queue size and ERED adjusts the drop rate.

It has been also shown in [1] that ERED enhances the performance of RED by reducing the drop rate. Congestion control literature questions the use of the actual queue size for congestion detection [29] [33]. Also the configuration and number of queue sectors are problematic in ERED. Erroneous configuration for these sectors can reduce network performance. Chapter 3 highlighted some of the shortcomings associated with ERED. A new strategy called Queue Sectors RED (QSRED) was proposed in Chapter 3 to enhance the functionality of ERED. Through simulation, QSRED shows preference over RED and ERED. However, the mismatch is still problematic and packets are unfairly dropped in the gateway.

4.2. Lock Out and Full Queue problems

As illustrated in previous chapters, congestion control in TCP networks consists of two approaches. The first approach is congestion recovery and the second is congestion avoidance.

Congestion recovery works after the gateway is overloaded and a packet is already dropped at the gateway. Strategies that use this approach are source algorithms. These algorithms adjust the sending rate upon congestion signals, such as triple acknowledgments, time-out or Explicit Congestion Notification (ECN). TCP Tahoe [38], TCP Reno [67], and TCP Vegas [8] are examples of these strategies.

Source algorithms operate end-to-end to send packets at the perfect rate using the congestion window size (*cwnd*). Additive Increase/Multiplicative Decrease (AIMD) is a technique to adjust the sending rate by modifying the

congestion window size. When a transmitted segment is successfully acknowledged then the window is additively increased. Conversely, the window is decreased in a multiplicative manner upon packet loss signal.

TD uses a technique to drop packets depending on the actual queue size. This technique has two main problems which are: the Lock Out and Full Queue problems. The first problem occurs when a few connections monopolize the whole queue space. The second problem occurs when the gateway keeps sending full queue signals to the sources for a long period of time [64]. Hence, there was a need to fix these two drawbacks of TD strategy using congestion avoidance approach.

In the congestion avoidance approach, some arrangements are made before the gateway is overloaded. The strategies that apply this approach are called network algorithms because they are applied by network components. Active Queue Management (AQM) is one of the algorithms that implements this approach [7]. Random Early Detection (RED) and its variants are the most popular strategies that adopted the AQM algorithm for congestion avoidance [29].

The congestion recovery and congestion avoidance algorithms work together to handle congestion. This Chapter questions the extent to which RED and the derivative approaches actually fixed the lock out and full queue problems. This Chapter also questions the cost of this solution.

4.2.1. *Early Congestion Control Strategies*

Traditional congestion control policies including; Source Quench, Fair Queuing, No Gateway Policy and Congestion Indication drop packets in the order they arrive. This is similar to the TD strategy and has similar problems, like full queue and lock out [33]. Aggressiveness against bursty traffics is another problem that result from these classic congestion control strategies [29] [33]. This happens because burstiness is one of the TCP traffic charac-

teristics and bursty connections always need more buffer size to absorb their traffic. In addition, these connections always try to monopolize more than their permitted share of bandwidth and buffer size. When a TCP window of n packets arrives at a TD congested gateway, all the packets of this window will be dropped in the order they arrive. Consequently, n congestion signals will be sent to the same source; lowering the sending rate of that connection.

Random Drop (RD) was designed by IETF to avoid these shortcomings of TD. The method of RD is to drop packets randomly rather than from the tail of a queue. In RD, the probability of a TCP flow losing a packet is proportional to the percentage of packets currently occupying the buffer. RD also has a number of shortcomings. It chooses packets to be dropped by inspecting the buffer distribution only at the time of overflow, disregarding all previous history. Therefore, it unfairly favors the connections with large packets [33].

Early Random Drop (ERD) strategy was designed to fix the problems of RD [33]. At imminent congestion, the gateway begins to drop packets at a rate that is derived from the current network congestion level. If the queue length is greater than the drop level – which is the threshold in RED – then ERD chooses packets randomly. If the probability of this packet is less than a preset drop probability then the packet is to be dropped.

4.2.2. *RED-Based Congestion Control Strategies*

To avoid inspecting the buffer distribution only at the time of overflow, which is a problem of RD, Random Early Detection (RED) inspects the average queue size for the previous history. Also, RED keeps two threshold parameters rather than one in ERD. In addition, the drop probability is dynamically adjusted during the network operation time.

The network flows which react to congestion, such as TCP flows, are responsive flows. Flows that do not adapt the sending rate based on conges-

tion conditions are unresponsive; such as UDP flows [71]. Unresponsive flows can occupy more than their allowed share of network resources, which causes aggressiveness against TCP connections. Thus, some RED variants were proposed to avoid this problem.

Flow Random Early Detection (FRED) preferentially discards packets from responsive and unresponsive flows. RED-DT [45] and Logest Queue Drop (LQD) [69] are per-flow strategy that distributes the buffer space fairly between responsive and unresponsive flows.

Dynamic And Self-Adaptive TCP-Friendly Congestion Control Mechanism (DATFCC) [73] adjusts the drop probability relating to the type of flow and the buffer usage ratio. It uses the TCP friendly approach [57], [5] to maintain fairness between TCP flows and real-time UDP flows. RED Optimized Dynamic Threshold (RED-ODT) uses the DT Scheme for shared buffer management [9] to adjust the maximum threshold and minimum threshold relating to the actual queue size and the buffer size in multi-queue gateways. Adaptive-RED (ARED) increases and decreases the max_p parameter. When the average queue size is greater than the maximum threshold, max_p is increased. When the average queue size is less than the minimum threshold, max_p is decreased [15].

The previously mentioned strategies attempt to increase network performance and avoid unnecessary drops in the gateway queue. However, because of the mismatch between the average and actual queue sizes, unfair drops are expected in RED gateways [64].

4.3. Adaptive RED

Adaptive RED (ARED) [25] [15] claims that there are two problems associated with RED's proposal. First, congestion control in RED does not keep track of the number of connections multiplexed across the link. Second, the benefits of RED are significantly reduced due to the packet drop technique that is used to indicate congestion.

Early detection of congestion in RED depends on the rate at which congestion notification is provided to the source and the network load. In the case of aggressive congestion notification, packet loss can be ignored at the price of low link utilization. If the congestion notification rate is conservative, link utilization will be increased at the price of high packet loss rate.

ARED proposed an ECN technique that is cognizant to the number of active connections multiplexed. The proposal provides an auto-configuration mechanism that determines the optimal parameters based on the traffic load. The strategy helps the gateway reduce packet loss and increase link utilization [25]. In addition, ARED maintains the main benefits of RED without any per-flow accounting. However, this additional functionality comes at the cost of additional complexity.

The main goal of ARED is to avoid packet loss due to buffer overflow. In the presence of heavy congestion, ARED marks packets from a large number of sources to reduce the offered load. Excessive congestion notification leads to under-utilization of the network bandwidth. Chapter four proposes the Risk Threshold RED (RTRED) mechanism, which provides techniques to reduce packet loss caused by buffer overflow or arbitrary congestion notifications. The Chapter also argues that the aggressiveness of ARED causes global synchronization.

According to ARED proposal [25], a 10 Mb/s capacity of a bottleneck link which is equally shared amongst 100 TCP connections will be reduced to 9.95 Mb/s just after receiving a congestion notification at one connection. If only two connections share the link then the offered load will be reduced to 7.5 Mb/s. Generally, If the bottleneck link is shared between n connections then one congestion signal arrives at any connection will reduce the offered load by $(1 - 1/2n)$. Thus, large n values reduce the impact of individual congestion notification. Therefore, RED parameter configuration should be more aggressive to prevent degenerating into a simple tail drop queue. Small values of n will maximize the impact of individual congestion notification,

TABLE 4.1. Adaptive RED algorithm.

Every avg update:
if ($min_{th} < avg < max_{th}$)
Status = Between;
if (avg $< min_{th}$ && status !=Below)
status = Below;
$max_p = max_p/\alpha$;
if (avg $> max_{th}$&& status !=Above)
status= Above;
$max_p = max_p{}^*\beta$;

resulting in under-utilization.

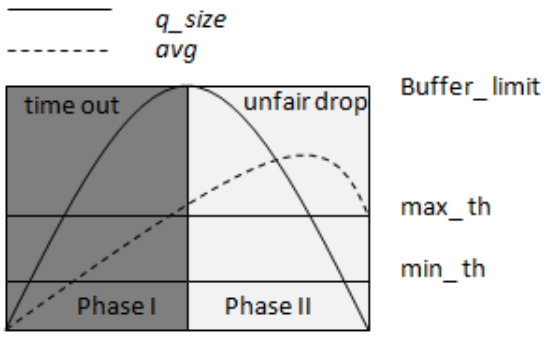
ARED parameter configuration is based on the explicit congestion notification and number of active connections. Table 4.1 illustrates the Adaptive RED algorithm. ARED examines the variations of the average queue size and makes decisions about when it should be less or more aggressive. If the average queue size is less than the minimum threshold then the maximum drop probability parameter max_p is scaled by the constant α. If the average queue size is greater than the maximum threshold then the parameter max_p is scaled with the constant β. The ARED proposal recommends the use of values 3 and 2 for α and β respectively.

4.4. RTRED strategy

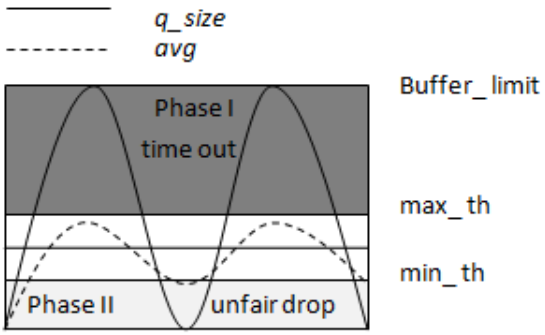
4.4.1. *RTRED Motivations*

New congestion control strategies tend to assign the congestion level for traffic management and congestion recovery processes. Rather than using the actual queue size as the congestion level indication, like TD strategy does, RED and some RED-based strategies keep an Exponentially Weighted Moving Average (EWMA) queue size to detect the congestion level.

RED was initially designed to minimize packet loss and queuing delays. It was also designed to maintain high link utilization and to remove biases against bursty traffic. Furthermore, it tries to avoid global synchronization which occurs when all network resources reduce their sending rate simultaneously.



(A) scenario I.



(B) scenario II.

FIGURE 4.1. Time out and unfair drop scenarios.

The problem with RED is the mismatch between the macroscopic and microscopic behaviors of queue length dynamics. This malfunction occurs when a peak in the actual queue size (microscopic behavior) exceeds the available buffer size. The average queue size is still low (macroscopic behavior) whereas the actual queue size is very high. This could happen because of small weight parameter w_q. As a result, a congestion signal is detected by network sources due to time outs caused by packet drops in the router. This means that TCP source algorithms, such as Reno, are responsible for congestion handling, whereas the network algorithm, which is a RED-based strategy, behaves like TD. This problem represent a mismatch between microscopic and macroscopic behavior of queue length dynamics.

Figures 4.1a and 4.1b illustrate the two possible scenarios of this mismatch. In phase 1, packets will be dropped due to time out signal. In phase 2 packets will be dropped unfairly because of the high value of avg parameter whereas the actual queue size is less than the min_{th}.

Choosing the drop level and the drop probability is the greatest challenge of RED and RED-based strategies. ARED proposed to multiply the max_p parameter by another parameter; α, when avg exceeds the parameter max_{th}, and to divide it by β when avg is less than the parameter min_{th}. An additional strategy named Gentle RED proposed to vary the drop probability from max_p to 1 when the average queue size varies from max_{th} to twice the max_{th} parameter [21]. Many RED variants have been proposed and the mismatch between the microscopic and macroscopic behaviors persist. In the next section, the Risk Threshold RED (RTRED) strategy is proposed which provides more timely congestion indication. RTRED reduces the amount of unfairly dropped or timed out packets.

4.4.2. *RTRED Algorithm*

The drop level defines the safe area of traffic fluctuation. Hence, drop level should alert the aggressive connections to slow down before the gateway is overloaded. It should be small enough for sources to respond before gateway overflow. Conversely, small threshold values would cause false congestion panic and unnecessary losses. Therefore, the threshold must be dynamically readjusted depending on the current network traffic.

The drop probability also, should be chosen carefully. It should be large enough to detect the misbehaved connections and small enough to protect the well behaved ones. Therefore, the drop probability should be adjusted dynamically as well.

TD strategy defines the drop level depending on the actual queue size. In addition to the global synchronization phenomenon, this scenario would lead to the Lock Out and Full Queue phenomena. RED strategy uses only the weighted average parameter to define the drop level. This sometimes leads to unfairly dropped or timed-out packets. In order to remedy the shortcomings of RED and TD strategies, RTRED uses both the actual and average queue

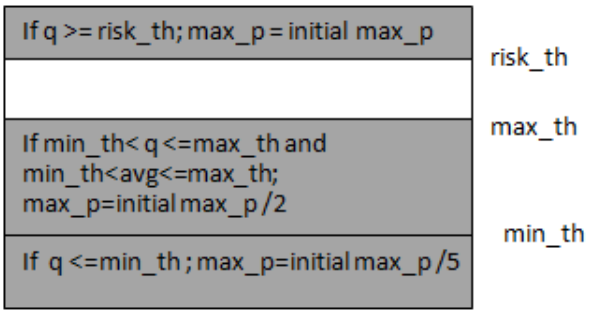


FIGURE 4.2. RTRED algorithm.

sizes to define the drop level.

Fig. 4.2 illustrates the proposed RTRED algorithm. In this algorithm, an extra drop level has been added, which is the risk threshold. If the actual queue size exceeds this risk threshold then RTRED uses the initial max_p parameter to calculate the drop probability. This will reduce the number of packets lost due to timeout signals. If the actual queue size is less than the min_{th}, then max_p parameter is reduced five times the initial max_p. This in turn will minimize the amount of unfairly dropped packets at the gateway.

When the average and actual queue sizes are in between the maximum and minimum thresholds, RTRED halves the parameter max_p. This allows the queue to safely increase without any risk of congestion. By doing so, RTRED reduces packet losses and better utilizes the queue space. It also removes aggressiveness against short lived bursty traffic.

4.5. Network topology

The simulator uses a network topology with six nodes sharing the same bottleneck link. The start time for nodes is uniformly distributed between 0 and 7 seconds with a 552 bytes packet size. The link between each node and the gateway is a duplex link with $10Mb/s$ capacity and a uniformly distributed delay between $1ms$ and $5ms$. The bottleneck link between the gateway and the sink is a duplex link with $0.7Mb/s$ capacity and $20ms$ delay. Fig. 4.3 illustrates this network topology.

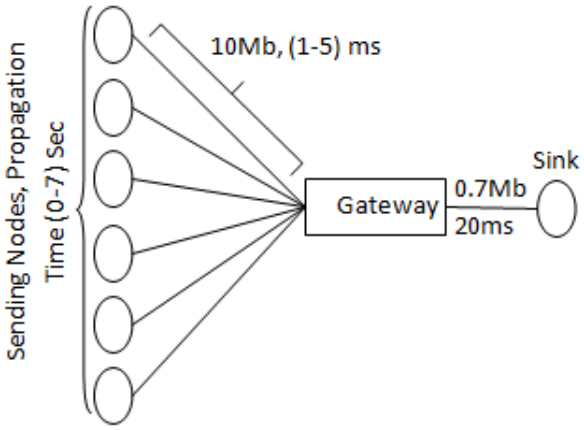


FIGURE 4.3. Network topology.

TABLE 4.2. Simulation and analysis (drop levels in packets)

Scenario	min_{th}	max_{th}	max_p	$risk_{th}$	Buf
1	12	25	0.05	28	30
2	60	90	0.10	95	100
3	15	30	0.08	40	50

4.6. Simulation and analysis

NS-2 script is used to simulate three different scenarios for RED, ARED and RTRED. Table 4.2, illustrates the parameter configuration for these scenarios.

Figure 4.2 divides the gateway queue into four areas. The white area in the table depicts the safe area in which the queue can fluctuate safely without any congestion indicator to absorb bursty traffic. The area between the $risk_{th}$ and the buffer limit is the time out area in which the risk of a packet drop due to time out signal is very high. The normal drop area is the area between the min_{th} and max_{th} parameters. RED and its variants normally start dropping packets in this area. Finally, packet drops in the area between the empty queue and the min_{th} are referred by unfair drops; hence the area is called the unfair drop area.

Figure 4.5 traces the percentage of packets that have been dropped in every area for each scenario separately. It shows that the majority of the packets have been dropped in the normal area and the time out area. Also,

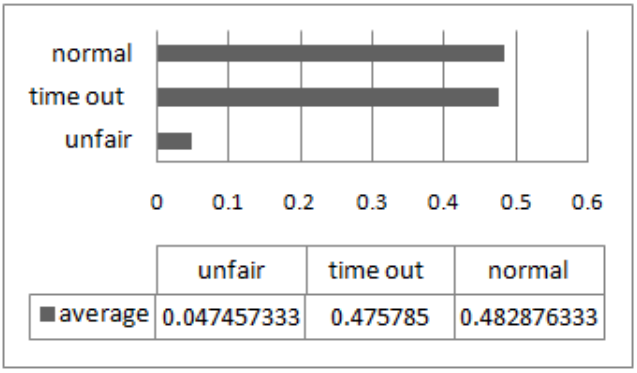


FIGURE 4.4. Total percentage of packets dropped for three scenarios.

some packets have been dropped in the unfair area. That motivates defering the use of the high initial max_p parameter until the actual queue size exceeds the $risk_{th}$, in RTRED. The bar chart in Fig. 4.4 depicts the average percentage of packets dropped for the three scenarios.

To avoid the mismatch between the microscopic and macroscopic behaviors of queue dynamics, RTRED works as a compromise between TD and RED strategies. It only checks the actual queue size in the Timeout and Unfair areas to avoid buffer overflow and unfair packet losses respectively. In order to reduce the drop rate and avoid wasting the queue space, it checks the average and actual queue sizes in the normal area, controlling congestion by reducing max_p to half of the initial parameter.

Table 4.3, illustrates the network performance for the three scenarios. It is clear from the table that RTRED strategy has outperformed RED and ARED strategies. It maintains a higher average queue size; increasing the gateway queue utilization. In response, this will reduce the packet loss rate which is not conflicted with the small average delay time values for RTRED. This technique reduces the total amount of packets propagated by the source nodes which in turn reduces the overhead of packet retransmission. The average delay time is calculated for the actual amount of packets queued at the gateway rather than the average queue size value.

Figures 4.6 and 4.7 illustrate scenario one simulation results. Fig. 4.6, depicts the drop probability and max_p parameters for the three strategies.

TABLE 4.3. Network performance for the three scenarios.

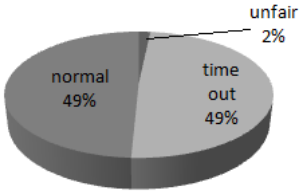
SCENARIO	STRATEGY	AVE Q SIZE (PACKETS)	AVERAGE DELAY (MS)	LOSS RATE (PACKETS)	NO. OF PROPAGATED PACKETS
	RED	20.2920	122.851840	710	18158
1	Adaptive	16.4690	123.832860	855	18303
	RTRED	21.2004	122.594742	668	18120
	RED	54.4667	119.340434	151	17639
2	Adaptive	60.9756	119.387794	147	17646
	RTRED	62.1799	119.259245	136	17627
	RED	21.0212	122.527085	651	18110
3	Adaptive	19.8217	122.804480	698	18151
	RTRED	24.8899	121.816685	548	18005

In this figure, the max_p parameter for RTRED fluctuates between the initial max_p parameter and $max_p = 5$ value, depending on the queue size dynamics. In response, the drop probability increases and decreases dynamically to fit the current congestion level. On the other hand, ARED keeps high max_p parameter which is unresponsive to the current congestion level. This is illustrated in Fig. 4.6a. RED also keeps a fixed max_p parameter, which is unresponsive to the queue dynamics as Fig. 4.6c shows. Figure 4.7 plots the average and actual queue sizes for the three strategies.

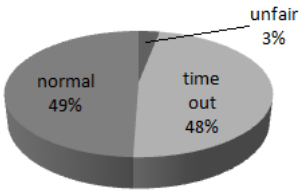
In Fig. 4.7b, RTRED strategy allows the queue to increase with a lower dropping rate. It stabilizes the queue size at the safe area with no risk of unnecessarily drop due to time out signal. Figures 4.7a and 4.7c are the figures of ARED and RED queue dynamics respectively. The figures show how aggressively and unfairly these strategies drop packets to avoid buffer overflow and to minimize the average queue size.

In this scenario, RTRED provides accurate calculation of the drop level and the drop probability. In fact, it reflects high trustworthy congestion indication before network starts recovering.

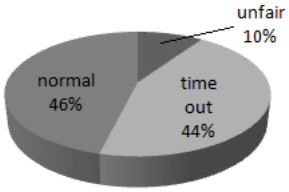
All AQM strategies involve tradeoffs. The tradeoff between decreasing delay and increasing throughput is one of the major problems that appears at the time of configuring the drop thresholds. Another tradeoff between link



(A) Scenario I.



(B) Scenario II.



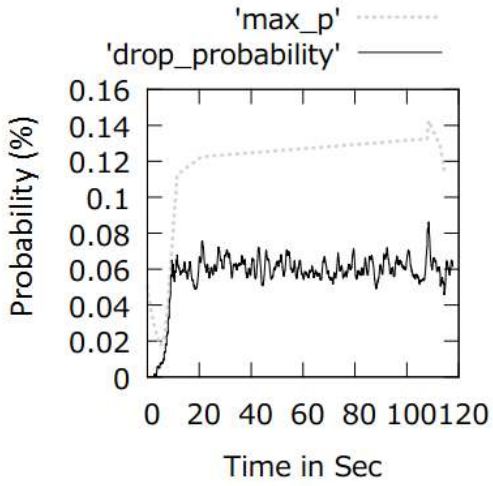
(C) Scenario III.

FIGURE 4.5. Percentage values for packets dropped in each drop area.

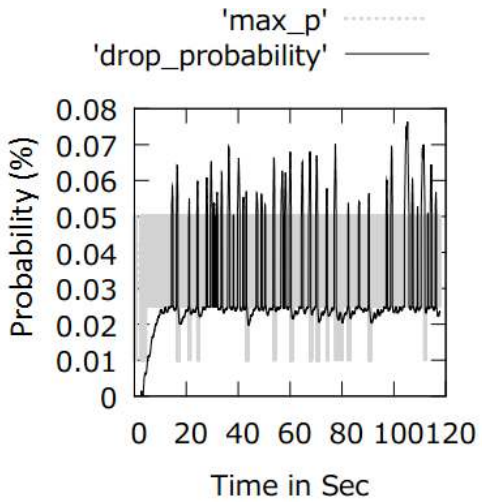
utilization and average delay time will worsen the problem. Larger min_{th} values will increase link utilization which is a good performance behavior, but in [29] it is suggested to set max_{th} twice the min_{th} which will increase delay.

The idea of small queues to reduce delay is proposed in [29]. RTRED can be sub-optimal, because, the desirable queue size is the size that helps the gateway reduces the drop rate and the overhead of packet retransmission. The average and actual queue sizes of RTRED are higher than RED and ARED. The actual end-to-end delays of RTRED, shown in table 4.3, are lower due to fewer retransmissions.

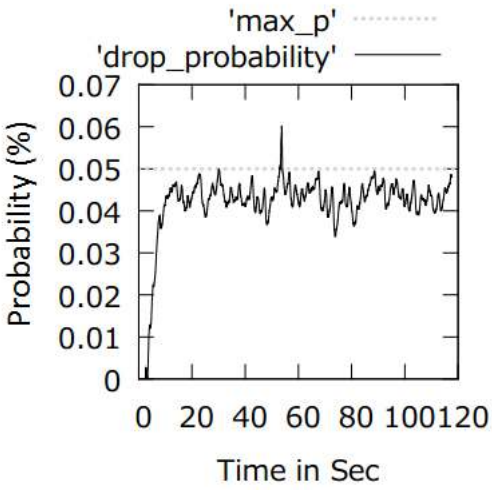
Figures 4.8 to 4.11 illustrate how RTRED outperforms RED and ARED for scenarios II and III respectively. The figures show the drop probability and the queue dynamics for each strategy. RTRED provides the most stable queue length with lower packet drops. It also maintains a dynamic max_p parame-



(A) Adaptive-RED.

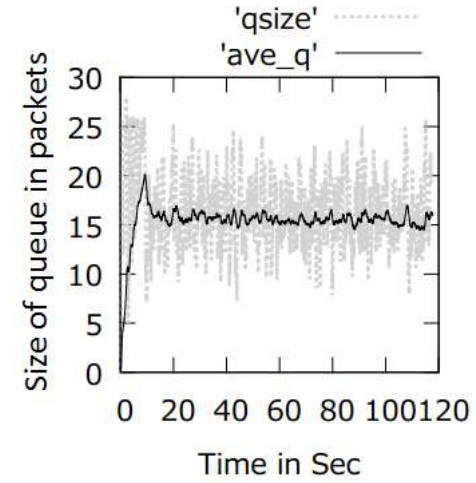


(B) RTRED.

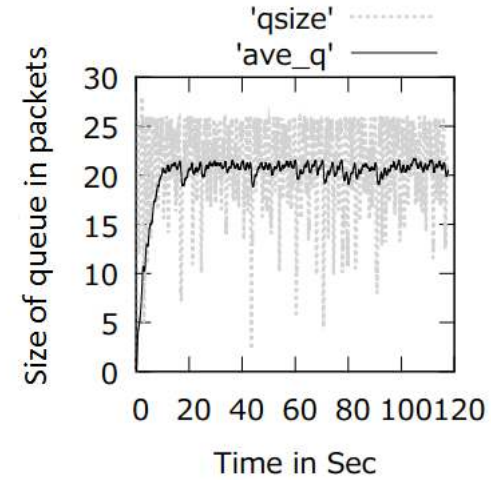


(C) RED.

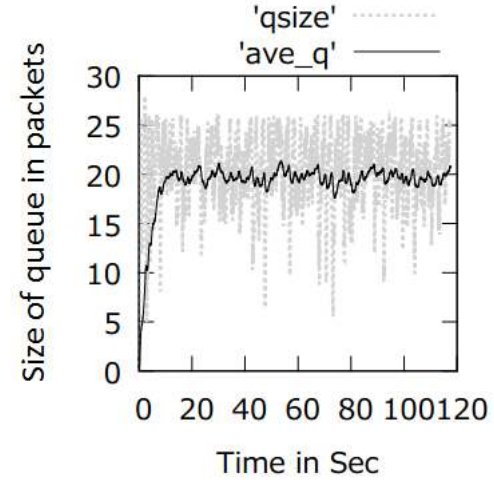
FIGURE 4.6. Drop probability and max_p parameters for scenario I.



(A) Adaptive-RED.

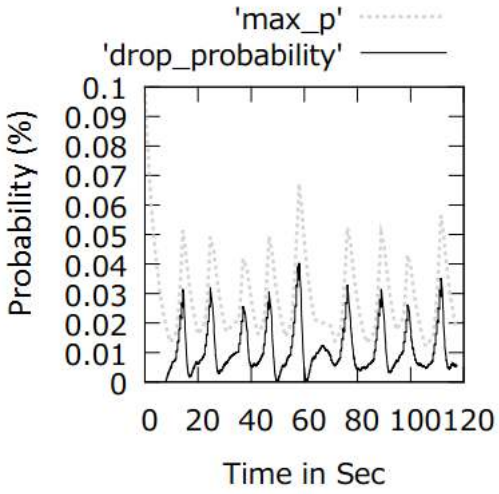


(B) RTRED.

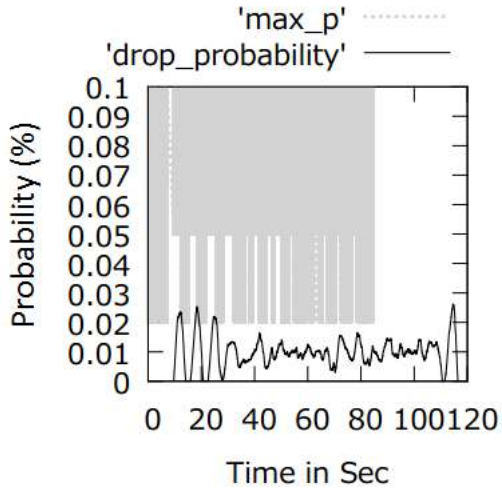


(C) RED.

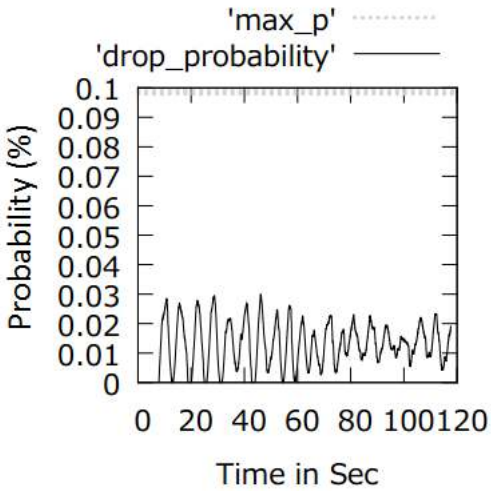
FIGURE 4.7. Average and actual queue sizes for scenario I.



(A) Adaptive-RED.

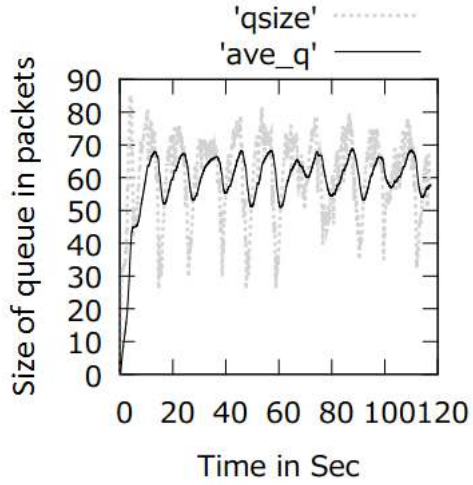


(B) RTRED.

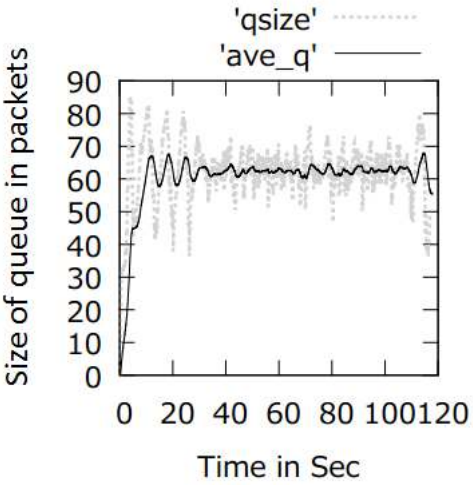


(C) RED.

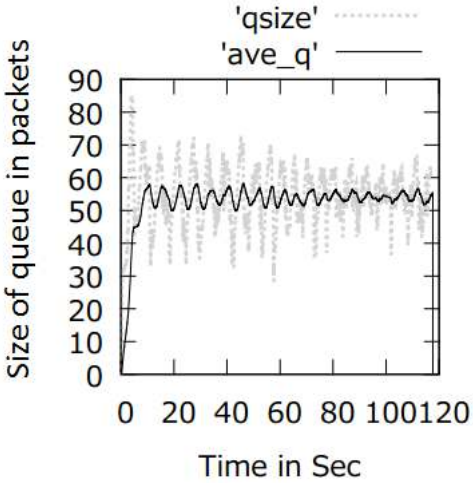
FIGURE 4.8. Drop probability and max_p parameters for scenario II.



(A) Adaptive-RED.

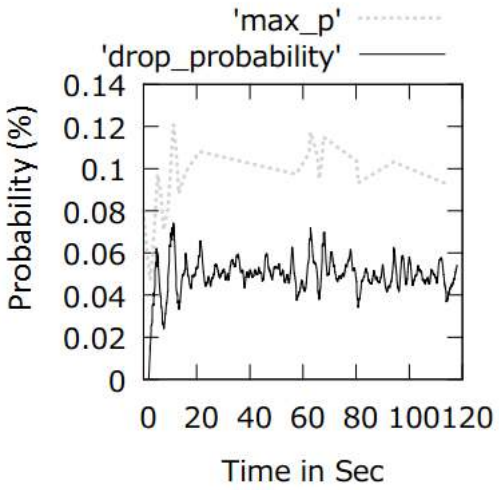


(B) RTRED.

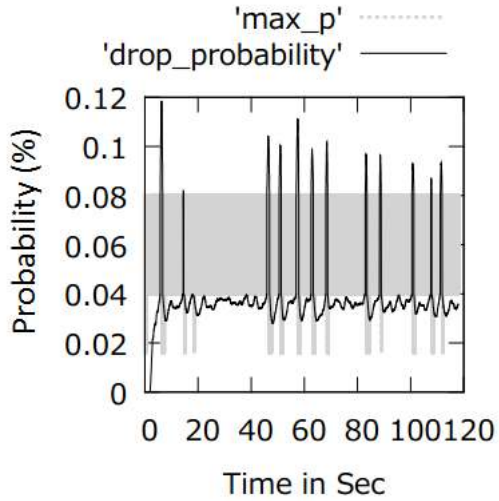


(C) RED.

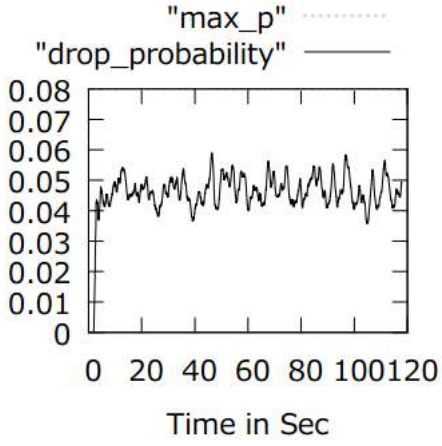
FIGURE 4.9. Average and actual queue sizes for scenario II.



(A) Adaptive-RED.

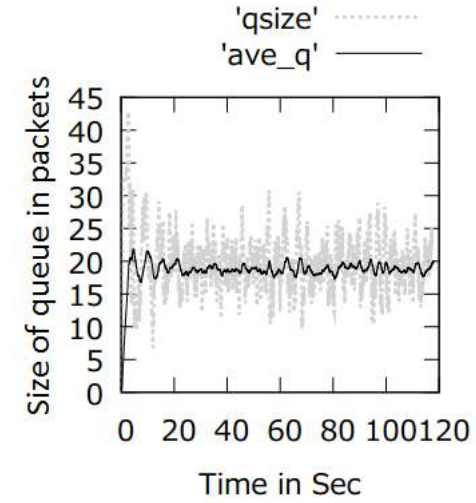


(B) RTRED.

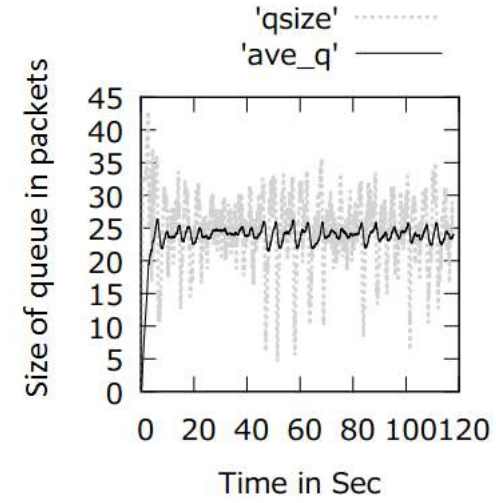


(C) RED.

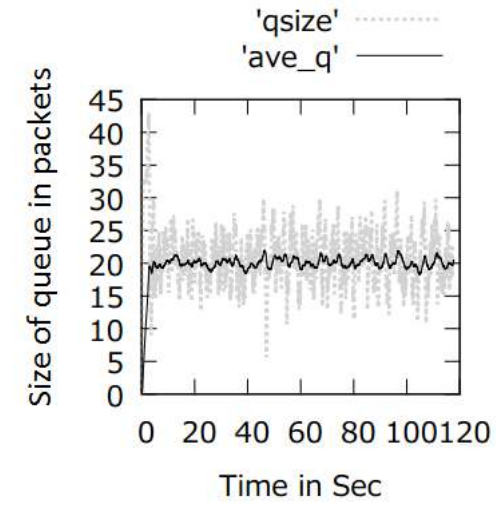
FIGURE 4.10. Drop probability and max_p parameters for scenario III.



(A) Adaptive-RED.



(B) RTRED.



(C) RED.

FIGURE 4.11. Average and actual queue sizes for scenario III.

ter and drop probability which are more responsive to the queue dynamics. RTRED does not increase the drop rate unless a strong signal of congestion is detected. Regardless of ARED's dynamic max_p, the adjustment of this parameter is problematic.

4.7. Summary

This Chapter presented a novel RED-based strategy; Risk Threshold RED (RTRED), which is designed to avoid the mismatch between the microscopic and macroscopic behaviors of queue length dynamics. The proposal is a compromise between the TD and RED strategies for congestion handling in TCP networks.

TD uses the actual queue size to define the congestion level. RED uses the Exponentially Weighted Moving Average to define the congestion level. RTRED uses both the actual and average queue sizes to calculate the drop probability and the congestion level. These calculations operate in conjunction with a third drop level; the risk threshold.

Using an NS-2 simulation, The results suggest that RTRED outperformed competing strategies; reducing unnecessary packet loss, delays and packet retransmissions. Furthermore, RTRED avoids wasting the gateway buffer size and increases the buffer utilization.

CHAPTER 5

RED Performance Enhancement Using The Weight Parameter

The mismatch between the average and actual queue sizes has been presented in previous chapters and the performance of ERED has been evaluated. Regardless of the improvement that has been achieved; the simulation results suggest that the actual queue size check could degrade the performace of RED and increase the complexity of the congestion control function. This chapter presents new technique to match both the average and actual queue sizes without applying the actual queue size check. This goal can be achived by maintaining adjustable weight parameter to increase the sensitivity of the average queue size to the changes in the actual queue size.

FRED is one of the few strategies that adjusts the weight parameter to avoid the mismatch between the average and actual queue sizes. This Chapter proposes the novel Weighted Random Early Detection (WTRED) strategy for congestion handling in TCP networks. The strategy dynamically adjusts RED's maximum threshold, minimum threshold and weight parameters to increase network performance. This Chapter also describes RED and FRED implementations and highlights their disadvantages. WTRED is compared with these classic congestion control strategies using an NS-2 simulation script. The simulation results demonstrate the shortcomings of RED and FRED. In addition, the results show that WTRED achieves greater link utilization and throughput than RED and FRED.

This Chapter is organized as follows: Section 5.1 gives a background about traditional congestion solutions. Section 5.2 reviews the performance of the Random Early Detection (RED) over TD mechanism. Section 5.3,

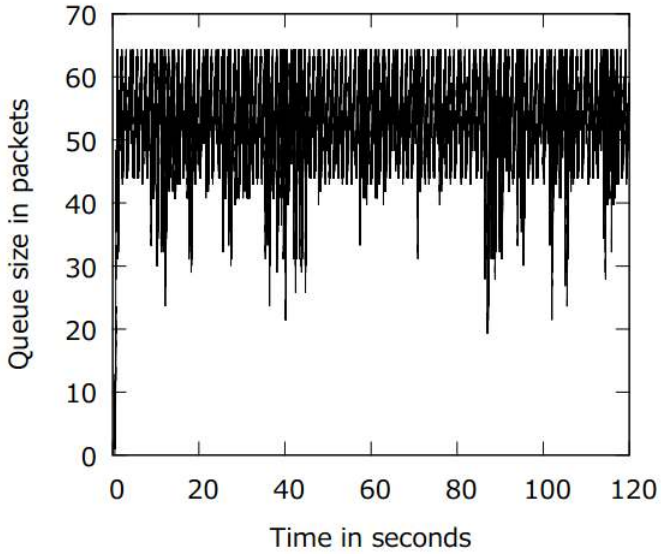


FIGURE 5.1. Full queue problem of TD strategy.

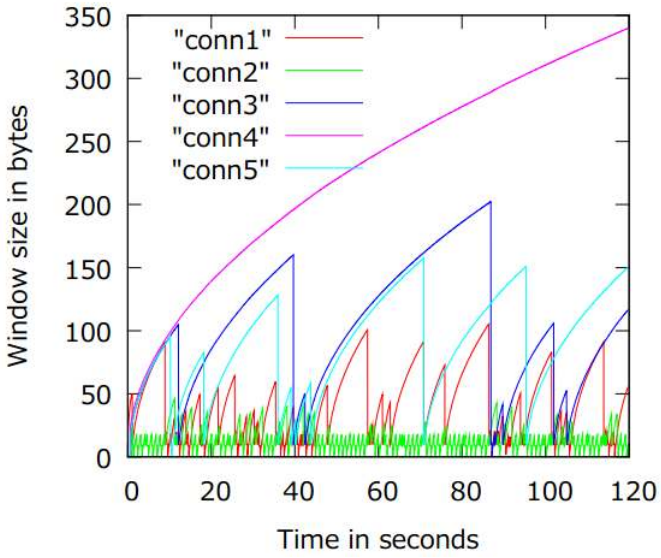


FIGURE 5.2. Lock out problem of TD strategy.

describes the Flow RED strategy. Refined Random Early detection (FRED) is detailed in section 5.4. Section 5.5 proposes the new weighted RED (WTRED) strategy. Simulation and discussion is presented in section 5.6; and section 5.7 summarizes the chapter.

5.1. Background

The performance of congestion control strategies is evaluated against the following goals [35]: (I) high bandwidth utilization, (II) fairness (III) reduced

jitter (IV) high responsiveness. (V) Fairness and compatibility with widely used protocols. Network performance involves four main parameters, which are: throughput, link utilization, packet loss and average delay. The design of a new congestion control strategy is subject to enhance one or more of these parameters.

The Tail Drop (TD) strategy uses a First In First Out (FIFO) queue management approach to control congestion. When the queue size reaches the buffer limit, packets are dropped in the order they arrive. This approach causes four problems that reduce network performance:

- Full Queue [64]: This problem occurs when a gateway continually sends full queue signals to sources for an extended period of time. In Fig. 5.1, a buffer of size 64 packets is full throughout the majority of the network operation time. In addition to the long delays associated with large queues, TD will inequitably drop packets, penalizing some connections. This will cause unfair resource allocation, which is illustrated in Fig. 5.2. In this figure, connection 2's window size is always lower than the other connections.
- Lock Out [64]: This problem occurs when TD allows a few connections to monopolize the whole buffer space. In Fig. 5.2, connection 4 receives more link bandwidth than the other connections in the network.
- Global Synchronization [33]: This problem occurs when all TCP senders reduce their sending rate simultaneously, reducing the network throughput [66] [80]. Fig. 5.3 shows TD algorithm causing global synchronization. Fig. 5.4 shows 5 seconds of global synchronization for the same scenario between time 20s to 25s.
- Bias against bursty traffic [28]: The nature of TCP transactions will often result in data packets being sent and received in groups. When the source receives acknowledgment of packet delivery from the desti-

nation node, it increases the number of packets to be sent in the next group. The maximum number of packets allowed in transit is the congestion window size *cwnd*. In network steady state, the congestion window size is increased. When the network becomes congested, the congestion window size is decreased. Due to the bursty nature of many transactions, dropping packets in the order that they arrive is unfair because it is likely that the majority of packets being dropped, may be from the same source. This will unfairly decrease the sending rate of that source whilst it needs more bandwidth to send this bursty traffic. Resulting in low network throughput.

Another strategy, Early Random Drop (ERD), begins packet drops at a rate derived from the current network congestion level. For example, if the queue size exceeds a certain threshold, then every arriving packet will be dropped with prefixed drop probability. The following code, illustrates the algorithm of this strategy:

Early Random Drop (ERD)'s Algorithm, see [33] for more details

if (queue_length $>$ drop_level) then

if get.random( ) $<$ drop_probablity then

drop(packet)

Random Early Detection (RED) was developed to maintain adjustable threshold and drop probability. RED solves the problems associated with the traditional congestion control strategies.

5.2. RED performance over TD

RED solves the problems associated with traditional congestion control strategies, such as TD. Fig. 5.5 depicts the weighted average and the actual queue sizes for the same scenario used in Fig. 5.1; in this instance RED rather than TD is used. Fig. 5.5 shows that RED has no biases against bursty traffic. While the average queue size is nearly 17 packets, a few bursts, between size 17 and 26, are allowed. The weighted average defines the level of congestion to

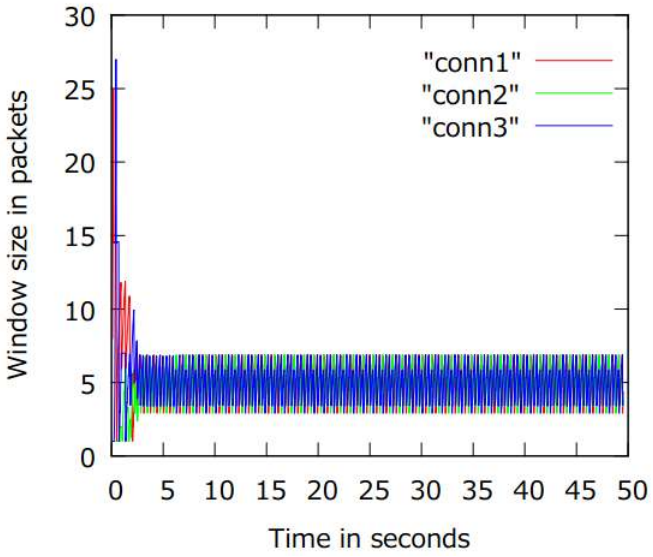


FIGURE 5.3. TD global synchronization.

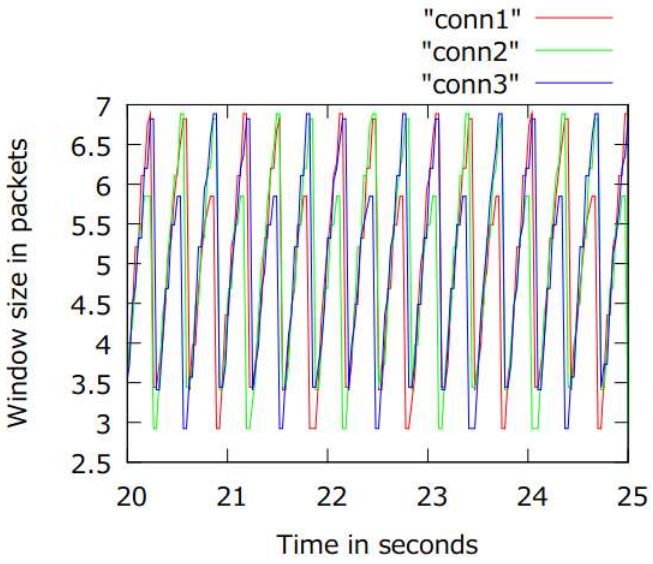


FIGURE 5.4. TD global synchronization (x 50 zoom in).

TABLE 5.1. The maximum drop probability for FRED's sub-phases.

Sub-Phase	1	2	3	4	5	6
max_p	max_p	$2max_p$	$4max_p$	$6max_p$	$8max_p$	$10max_p$

start packet dropping. In Fig. 5.5, avg is always less than 13 packets and this helps the gateway detect congestion before the buffer overflows. By reducing the actual and average queue sizes, RED lowers queuing delays, prevents the full queue problem and provides fair resource allocation.

RED also solves lock out and global synchronization problems. Fig. 5.6 shows the congestion window size for RED. It is the same network used to plot Fig. 5.2 with TD. It is clear from Fig. 5.6, that the connections receive fair resource allocation. Lock out problems are also eliminated. The global synchronization, caused by TD, is also solved. This is evidenced in Fig. 5.7.

Despite the strong design of RED, drawbacks have been revealed in a number of studies. Parameter configuration and the recommended values have been subject to change over time [44]. For example, ARED dynamically adjusts the max_p parameter depending on the values of avg , max_{th} and min_{th} [15]. Blue-RED is another strategy that dynamically adjusts the drop probability of RED p_a to increase network performance [16]. Other strategies suggest dynamic thresholds for RED, such as RED-DT [71] and PDT-RED [68]. Diffusion Early Marking (DEM) [6] is a RED based strategy designed to avoid global synchronization. In order to avoid excessive congestion marking, DEM has been designed to optimize the distribution of packet marking. For further details in newly designed congestion control strategies, see [74].

Parameter setting in new RED-based strategies is dynamic. Some studies suggest optimal values for the weight parameter w_q. BF-RED [76], proposes different weight parameters for different flows to ensure fair bandwidth allocation. Also, RT-RED adjusts RED's parameters based on the ratio between the actual queue size and the average queue size. RED performance evaluations in real networks are further investigated in [10] [43] [51].

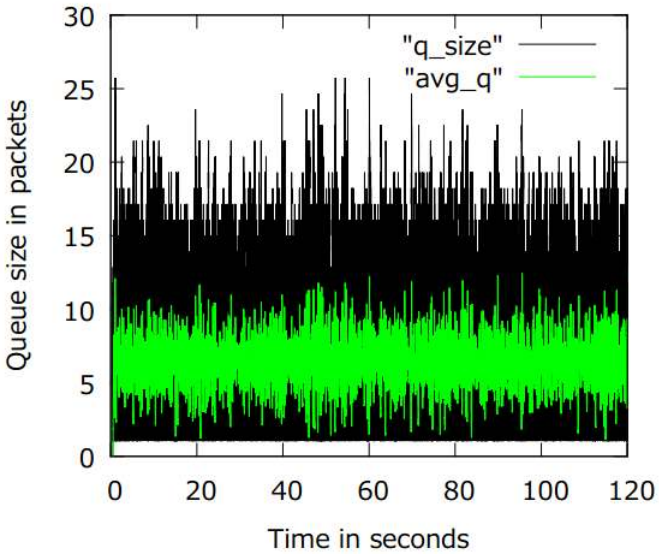


FIGURE 5.5. Average and actual queue sizes on a RED gateway.

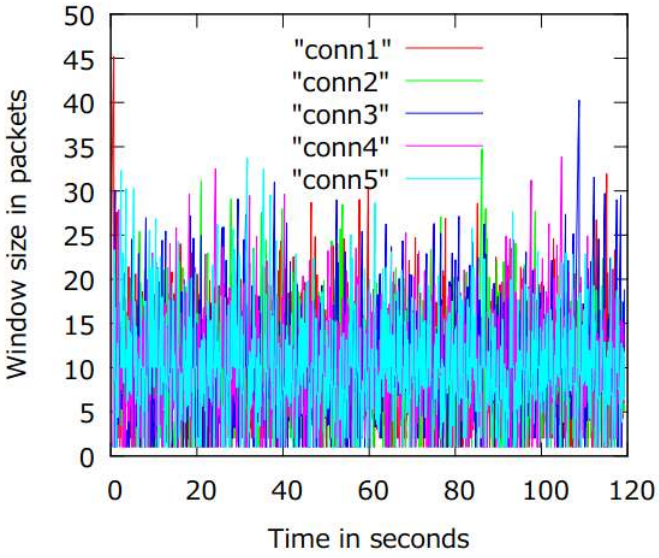


FIGURE 5.6. Congestion window size on a RED gateway.

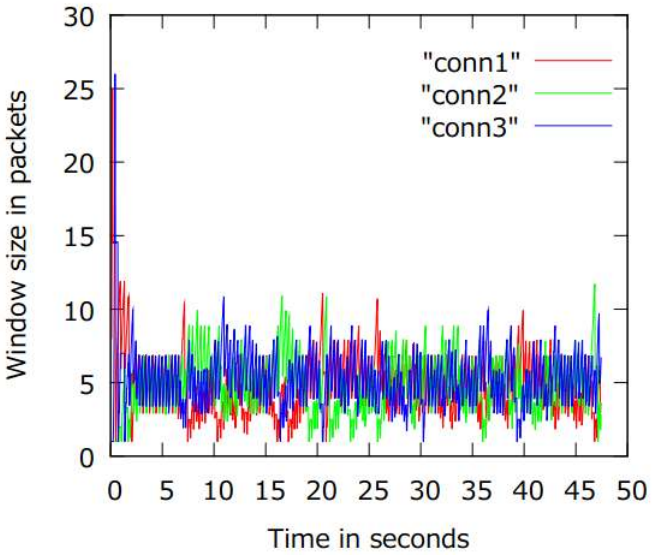


FIGURE 5.7. Congestion window size on a RED gateway without global synchronization.

5.3. Flow RED

One of the design principles of RED is to maintain low average queue size and allow fluctuations in the actual queue size in an effort to absorb transient congestion and bursty traffic. The Refined Design of RED (FRED) [72] dynamically adjusts the weight parameter w_q according to the change of the actual queue size. Another strategy Flow RED which is also denoted by FRED [44] focused on shared networks. This strategy provides fair resource allocation between responsive connections, such as TCP, and unresponsive connections, such as UDP. Responsive connections react to congestion signals by reducing their window size. Unresponsive connections do not respond to congestion signals. Thus, unresponsive connections will occupy more bandwidth than responsive connections. Fairness problem arising from unresponsive connections are outside the scope of this thesis. It is only described here to remove the ambiguity between the Flow RED (FRED) and the Refined Design of RED (FRED). For the rest of this chapter, the term FRED is used to refer to the refined design of RED.

Sub-Phase	1	2	3	4	5	6
w_q	w_q	$4w_q$	$8w_q$	$12w_q$	$16w_q$	$20w_q$

TABLE 5.2. The weight parameter for FRED's sub-phases.

5.4. Refined design of RED (FRED)

Refined Random Early Detection (FRED) [72] is a modication of RED that uses a dynamic weight w_q and max_p to control congestion. It divides the area between the maximum threshold and minimum threshold into six sub-phases. As the average increases from the minimum threshold to the maximum threshold, the max_p is increased. A different value is assigned for the different sub-phases illustrated in Table 5.1.

In addition to a dynamic max_p parameter, FRED maintains a dynamic weight parameter w_q. It, also, makes use of a third threshold, called the warn threshold. The weight parameter is adjusted whenever the actual queue size exceeds the warn threshold. FRED normally assigns half of the buffer size to this value. After the actual queue size exceeds the warn threshold, it has to go through another six sub-phases, but this time, with different weight parameters for each sub-phase. The weight values for these sub-phases are illustrated in Table 5.2. In order to apply the weight parameter sub-phases, the actual queue size must be greater than the average queue size.

RED calculates the average queue size for every packet arrival. FRED extends RED by additionally calculating the average queue size for every packet departure.

FRED provides a technique to avoid negative feedback of transient congestion. It proposes that transient congestion is harmful when the queue is almost full. In RED, the weight parameter w_q controls changes in the average queue size. This parameter is constant in the original RED algorithm. FRED adjusts the weight parameter dynamically according to the changes of the actual queue size. Reconfiguring w_q in FRED provides a timely manner to quench the actual queue size when the buffer is nearly full.

In addition, FRED dynamically adjusts the maximum drop probability max_p with respect to the average queue size. Even though max_p reflects the aggressiveness of the active queue management technique, RED maintains a fixed value for it. FRED uses the surplus between the actual and average queue size to indicate the aggressiveness of traffic. If the surplus is high, then the incoming traffic is bursty. If the surplus is increased suddenly then the incoming bursts exceed the gateway capacity and buffer overflow is expected.

FRED adds two enhancements to RED's function. The first enhancement is to adjust the weight parameter dynamically according to the changes of surplus between the average and actual queue sizes. The second enhancement by FRED is to estimate the average queue size with every packet departure.

This proposal enhances the technique of calculating the average queue size. Conversely, it could cause severe drawbacks in some situations [72]. The value of the weight parameter can be maximized to increase the sensitivity of RED to the initial stages of congestion and bursty traffic. However, quick increases in the weight parameter make RED very sensitive to the short-lived bursty traffic and transient congestion, which wastes the buffer space.

5.4.1. *FRED algorithm*

FRED adjusts the weight parameter w_q and the maximum drop probability max_p dynamically. The amount of burst traffic that can be accommodated by the gateway is bounded by the buffer size. Large buffers can accommodate large traffic bursts. The surplus between the average and actual queue size is estimated according to the buffer size.

FRED estimates the average queue size for every packet departure as opposed to every packet arrival. More precisely, FRED calculates the average queue size for both packet arrival and departure events. In some cases, this scenario does not work perfectly. If the queue management technique uses small value for w_q then the average queue size is decreased slowly. In this case

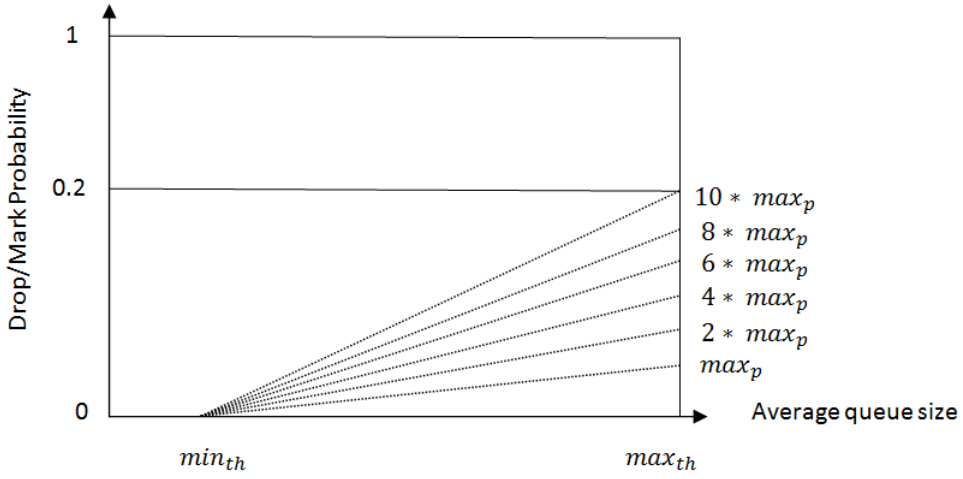


FIGURE 5.8. Dynamic adjustment of FRED's maximum drop probability.

with successive packet departures, the incoming traffic will be dropped due to the high average queue size, even though there is enough buffer space to accommodate this traffic.

A simple solution is proposed by FRED to overcome the previous problem. Upon packet departure, FRED compares the average and actual queue sizes. If the average queue size is smaller than the actual queue size then FRED recognizes that there is a disparity between packet arrival and departure rates. Thus, the active queue management technique of FRED increases the weight parameter to reflect the packet de-queuing. In this case, the drop probability, has to be decreased to reduce the drop rate of incoming traffic.

FRED algorithm consists of two parts. First, the dynamic adjustment of the weight parameter which is illustrated in Table 5.3 and Eq. 5.16. Second, the fine-grained setting of max_p which is illustrated in Fig. 5.8.

5.4.2. *Dynamic adjustment of the weight parameter*

The weight parameter in classic RED is a constant. If the weight parameter is too small then RED queue management technique would be too slow to react to extremely bursty traffic [72]. If the weight parameter is too large then the queue management technique will be too aggressive against short-lived

TABLE 5.3. Dynamic adjustment of FRED's weight parameter.

For very packet arrival:
$q++$;
if $q > avg$
$diff = q - avg$;
else $diff = 0$;
$ratio = diff/buff_s$;
R =int$(10 * ratio)$;
If $(q < warn - line)$
$new - w_q = old - w_q$;
else{
switch (R) {
case 0: $new - w_q = old - w_q$;
case 1: $new - w_q = old - w_q$ * 4;
case 2: $new - w_q = old - w_q$ * 8;
case 3: $new - w_q = old - w_q$* 12;
case 4: $new - w_q = old - w_q$* 16;
default: $new - w_q = old - w_q$ *20;
}
}
For very packet departure:
$q--$;
if $(avg > q)$
$new - w_q = old - w_q$* 10;
else $new - w_q = old - w_q$;
Where:
$old - w_q$:the initial weight parameter,default 0.02.
$buff_s$: buffer size.
$warn_{line}$:half of the buffer size.
q: actual queue size.
avg: average queue size.
$diff$: surplus of the actual queue size over the average queue size.
$ratio$: ratio of surplus over the buffer size.
R: the integer part of (10*ratio).
$new - w_q$: the calculated value for the new weight.

bursty traffic resulting in under-utilization.

Upon packet arrival, FRED adjust the weight parameter. The changes in the actual queue size are monitored. A new warning threshold is added to split the setting of the weight parameter into two phases. If the actual queue size exceeds the warning line then w_q is adjusted using Eq. 5.16.

$$new - w_q = \begin{cases} old - w_q, & R \in [0, 0.1) \\ old - w_q * 4, & R \in [0.1, 0.2) \\ old - w_q * 8, & R \in [0.2, 0.3) \\ old - w_q * 12, & R \in [0.3, 0.4) \\ old - w_q * 16, & R \in [0.4, 0.5) \\ old - w_q * 20, & R \in [0.5, 1] \end{cases} \tag{5.16}$$

Where:

R is the ratio of the surplus of the queue size to the buffer size.

$new - w_q$ is the new value of the weight parameter.

$old - w_q$ is the initial value of the weight parameter.

It is recommended by FRED to set the warning line to half of the buffer size. Upon packet departure, the weight parameter is set to 0.02. Thus high value of w_q makes the average queue size more sensitive to rapid decreases in the actual queue size. Table 5.3 illustrates this algorithm.

5.4.3. *Fine-grained setting of the maximum drop probability*

The maximum drop probability max_p is a very sensitive parameter. The aggressiveness of RED is determined by this parameter. FRED divides the area between the minimum and maximum thresholds into several sub-phases. Each sub-phase is assigned a different max_p value. When the average queue size increases from the minimum threshold toward the maximum threshold, the

maximum drop probability is also increased. The number of max_p sub-phases is bounded by the actual buffer size and the space between the minimum and maximum thresholds. It is recommended by FRED to set each sub-phase to 10% of the actual buffer size with six sub-phases.

RED does not adjust the max_p value. Hence, RED does not work effectively with different traffic load variations [72]. FRED provides fine-grained max_p setting which is illustrated in Fig. 5.8.

5.5. WTRED design guidelines

Before outlining the WTRED algorithm, it is better to highlight some of the parameter and implementation issues with RED and FRED.

5.5.1. *RED's parameters*

RED maintains a set of parameters to prevent congestion. Improper parameter configuration increases the oscillations in the average queue size and data packet loss [14]. Thus, RED suffers from heavy oscillations in the average queue size, which reduces the network stability. Some RED variants provide auto parametrization for RED [75], however, parameter configuration is problematic. The following section outlines the main parameters of RED:

Average queue size: Rather than using the actual queue to reflect congestion in gateways, RED uses an Exponentially Weighted Moving Average (EWMA) to reflect historical queue dynamics. This EWMA helps gateways distinguish between transient and permanent congestion. It also avoids global synchronization and bias against bursty traffic [29]. RED uses Eq. 2.3 to calculate the EWMA of the queue size.

Minimum threshold: The first dropping level of RED. When the average queue size is lower than this threshold, no packets will be dropped.

Higher minimum thresholds increase throughput and link utilization [29].

Current drop probability: The immediate drop probability in Eq. 2.4 which is used to calculate the accumulative drop probability in Eq. 2.5. As shown in Eq. 2.4, the current drop probability p_b is a function of the maximum drop probability max_p. Consequently, the accumulative drop probability p_a is a function of max_p. A higher max_p will result in a higher drop rate. Section 5.6.3, shows how this can cause problems with FRED's implementation.

Accumulative drop probability: This parameter is a value between zero and one. When the average queue size exceeds the minimum threshold, a packet is chosen randomly from the queue. If the probability is less than the drop probability then the packet is dropped. The calculation of this parameter is illustrated in Eq. 2.5.

Maximum drop probability: This parameter reflects the maximum drop rate of RED. For example, if the maximum drop probability is 0.1, the gateway cannot drop more than one packet out of ten which is calculated using the formula 1 out of $1/max_p$.

Weight parameter: This parameter is used to calculate the EWMA queue size in Eq.2.3. It reflects the sensitivity of the average to the actual changes in the queue size. It takes values from zero to one. Setting the weight parameter to larger values means that fewer packets are required to increase the average from A to B. For instance: if a RED gateway with a weight parameter of 0.001 needs 60 packet arrivals to increase the average queue size from 6 to 10, then the same gateway with 0.003 weight parameter will need fewer packets (40) to increase the average from 6 to 10.

Maximum threshold: As the average queue size increases from the minimum threshold toward the maximum threshold, the drop rate is increased slowly. As the average queue size hits the maximum thresh-

old, the drop probability turns to one and every arriving packet has to be dropped. This keeps the actual queue size between the minimum and the maximum threshold. In RED, the values of the minimum and maximum threshold are assigned depending on the desirable actual average of the queue size. The new strategy, WTRED, uses different criteria to configure these parameters. This is further detailed in Sec. 5.5.2.

5.5.2. *WTRED proposal*

The maximum and minimum thresholds in RED divide the buffer into three main areas. The area between 0 and the minimum threshold, the area between the minimum threshold and maximum threshold and the area between the maximum threshold and buffer limit. In FRED, these areas are referred by green, yellow and red respectively. When the average queue size is in the green area, all traffic is allowed with no drops. In the yellow area, some of the traffic will be dropped. In the red area, no traffic is allowed and all incoming packets will be dropped.

In RED, the maximum and minimum thresholds are dependent on the actual buffer size which is preset before network operation time. The weight parameter is also a prefixed parameter. The default values of the maximum threshold, minimum threshold and the weight parameter in RED are 5, 15 and 0.002 respectively. It has been suggested that the maximum threshold should be set to twice the minimum threshold [29].

In FRED, the green area is divided into six equal areas with different maximum drop probabilities. Also, the area between the warn threshold and the buffer limit is divided into another six equal areas, each area with a different weight parameter. The mechanism proposed in this study, WTRED, uses different weight parameters for each area in RED. Furthermore, WTRED adjusts the maximum and minimum thresholds based on the actual buffer size.

Figures 5.9 to 5.12 illustrate the network performance for the topology depicted in Fig. 5.14. NS-2 was used to examine the four network performance parameters with weight parameters in the range 0.001 to 0.1. The results suggest that RED works most efficiently when the weight parameter is between 0.001 and 0.003. To be more specific, two performance parameters are improved: throughput and link utilization. Loss rate will be at an acceptable level but delay will increase. These results agree with the original parameter recommendations for RED [29] that suggest a weight parameter of 0.002.

The new parameter configuration in WTRED is to assign the weights 0.003, 0.002 and 0.001 to the green, yellow and red areas respectively. Also, the minimum threshold will be set at 40% of the buffer size and the maximum threshold will be 70% of the buffer size. This high minimum threshold will grant higher network throughput and link utilization. For the average to respond quickly to the changes in the green area WTRED assigns the value 0.003 to the weight parameter. In case that persistent traffic bursts accumulate the queue size, the average will be increased faster to hit the minimum threshold and initiate congestion recovery.

When the average reaches the yellow area, RED starts dropping packets. In this area there is no need to have higher value for the weight parameter. Hence, WTRED assigns the value 0.002 to the weight parameter. Another reason to use a lower weight parameter in this area is to maintain reasonable time before the average hits the maximum threshold. When the maximum threshold is reached, the drop probability will be 1.0 and every arriving packet will be dropped. Setting the weight parameter to 0.002 in this area is better than using high values for the maximum drop probability. High max_p values lead to shorter time to reach the maximum threshold which will reduce the link utilization.

When the average queue size exceeds the maximum threshold and enters the red area, RED will drop every packet arriving at the gateway. FRED uses higher weight parameters to increase the sensitivity to the changes in the

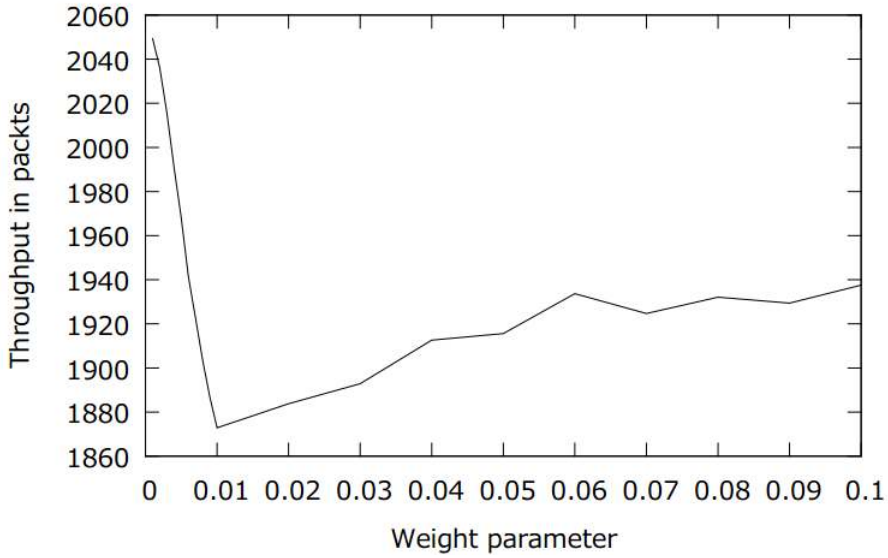


FIGURE 5.9. Throughput for a range of weight parameters from 0.001 - 0.1.

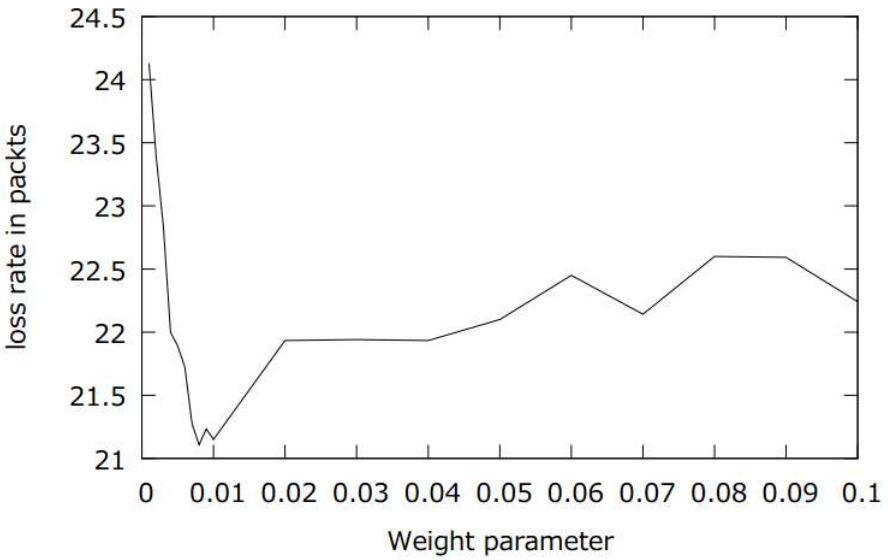


FIGURE 5.10. Loss rate for a range of weight parameters from 0.001 - 0.1.

actual queue size. In this case, the actual queue size and the average queue size will closely follow each other and FRED's behavior may approach the TD strategy, dropping packets based on the actual queue size. Fig. 5.13 illustrates the WTRED algorithm.

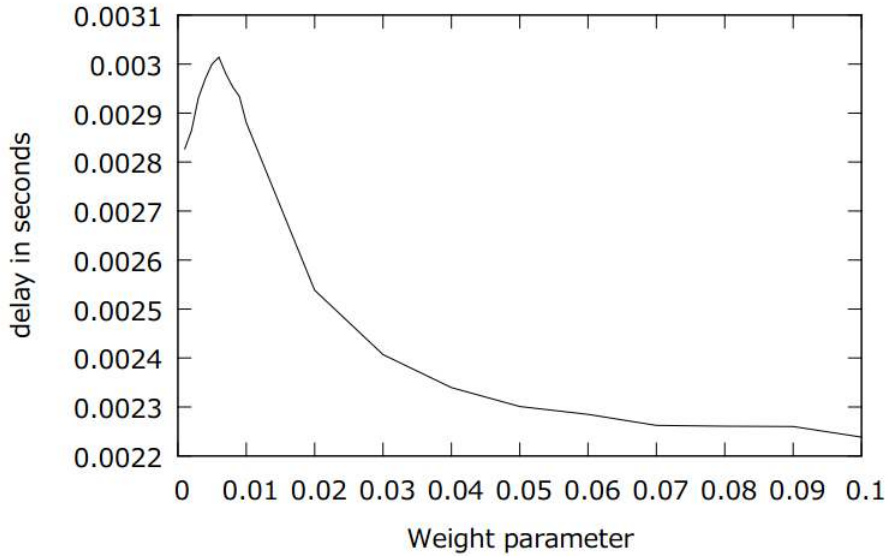


FIGURE 5.11. Delays for a range of weight parameters from 0.001 - 0.1.

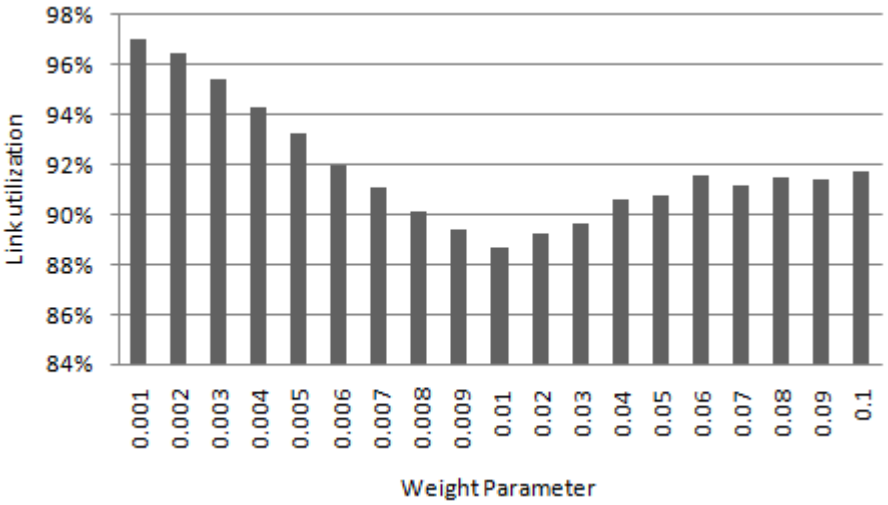


FIGURE 5.12. Link utilization for a range of weight parameters from 0.001-0.1.

5.6. Simulation and Discussion

WTRED is simulated using the NS-2 simulator. WTRED, RED and FRED are compared against the four network performance parameters which are: throughput, link utilization, average delay and packet loss. The network topology in Fig. 5.14 is used to run 11 different scenarios with different weight parameters and buffer sizes. Table 5.4, illustrates the weight and buffer size for each scenario used in this simulator.

w_q	0.001	0.0015	0.002	0.0025	0.003	0.0035	0.004	0.0045	0.005	0.0055	0.006
Buffer (packet)	40	50	60	70	80	90	100	110	120	130	140

TABLE 5.4. Buffer sizes for FRED's sub-phases.

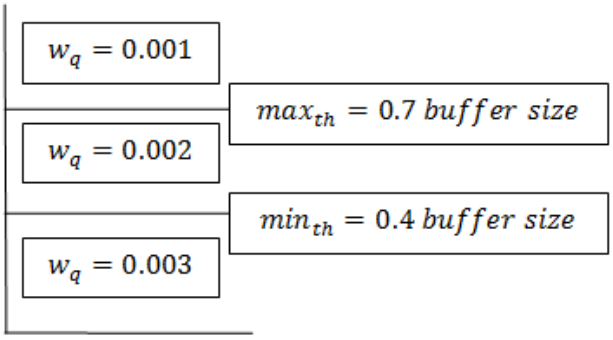


FIGURE 5.13. WTRED algorithm.

5.6.1. *Network topology*

NS2 simulation script has been used to define five FTP sources. A duplex link with $10Mb/s$ bandwidth connects each source with the gateway. Connection delays are uniformly distributed between $1ms$ and $5ms$. Another duplex link with $10Mb/s$ bandwidth and $5ms$ delay connects the gateway with a TCP sink. The packet size is 552 bytes and the TCP variant is Reno. Fig. 5.14 illustrates the simulation network topology.

5.6.2. *Simulation results*

RED [29] suggests that, in order to fillter out transient congestion at the gateway, the weight (w_q) must be assigned small values. Low weights mean that the EWMA will respond slowly to actual queue size changes. This reduces the gateway's ability to detect the initial stages of congestion. The maximum and minimum threshold values are restricted by the desired average queue size. Also, the difference between the maximum threshold and minimum threshold must be sufficiently large to avoid global synchronization. Small differences between maximum threshold and minimum threshold allow the average queue size to oscillate up to the maximum threshold.

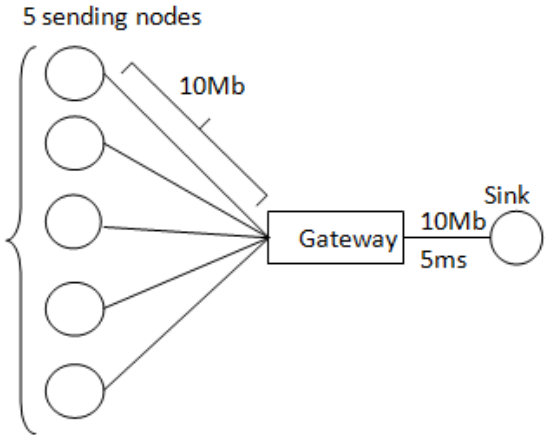


FIGURE 5.14. Simulation network topology.

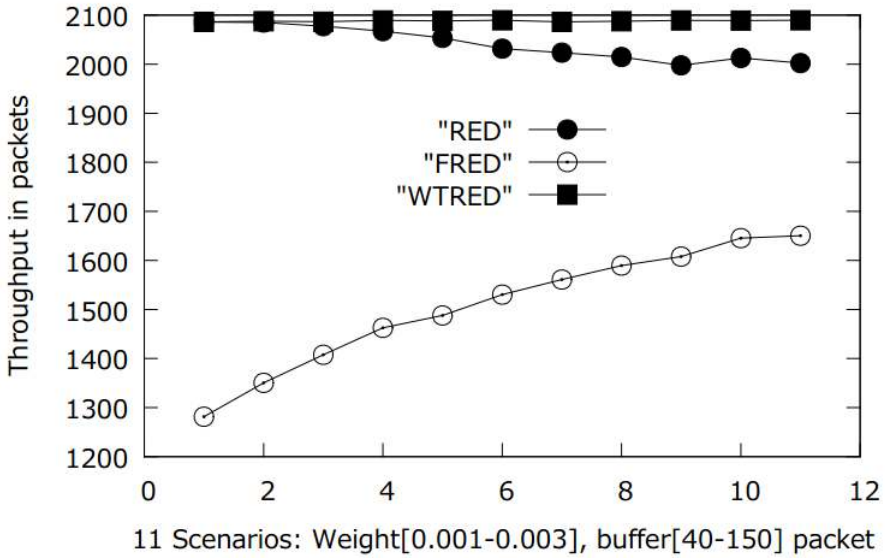


FIGURE 5.15. Network throughput for RED, FRED and WTRED.

Figures 5.15 to 5.18 depict the throughput, packet losses, average delay and link utilization respectively. Fig. 5.15 shows that WTRED achieved the highest throughput among the three strategies. FRED generates very poor throughput due to the high weight parameter and maximum drop probability. This also increases the loss rate as shown in Fig. 5.16. FRED also has the lowest delays among the three strategies as in Fig. 5.17. This comes at the price of very poor throughput, Fig. 5.15 and link utilization, Fig. 5.18.

The figures demonstrate that WTRED outperforms RED and FRED. WTRED improves throughput and link utilization while maintaining acceptable delays and loss rates.

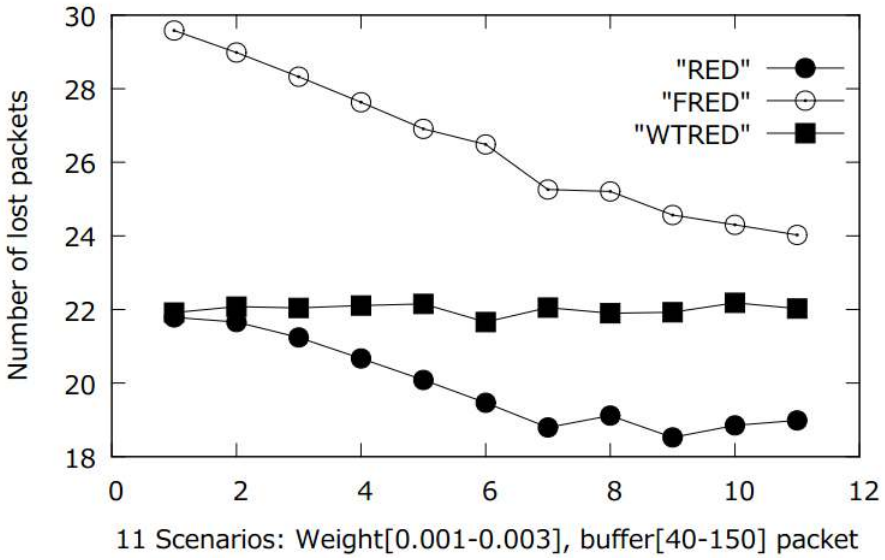


FIGURE 5.16. Packet loss rate for RED, FRED and WTRED.

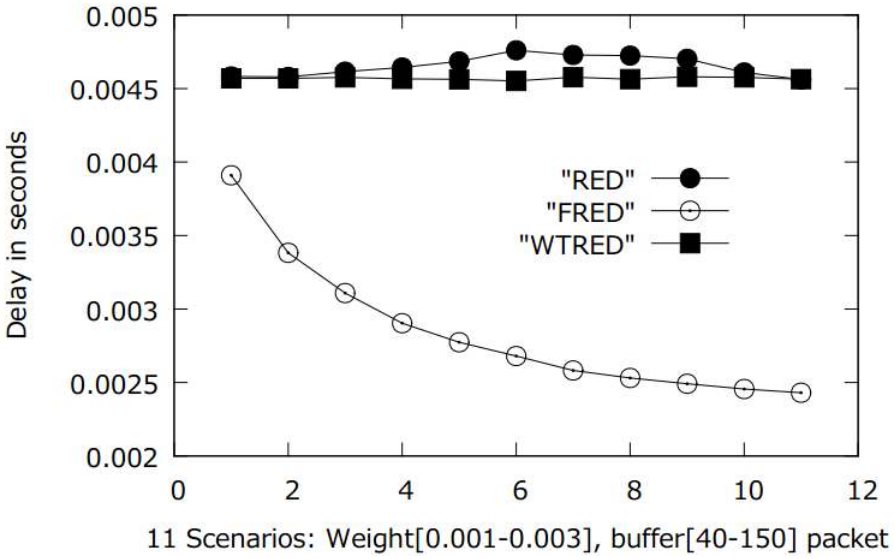
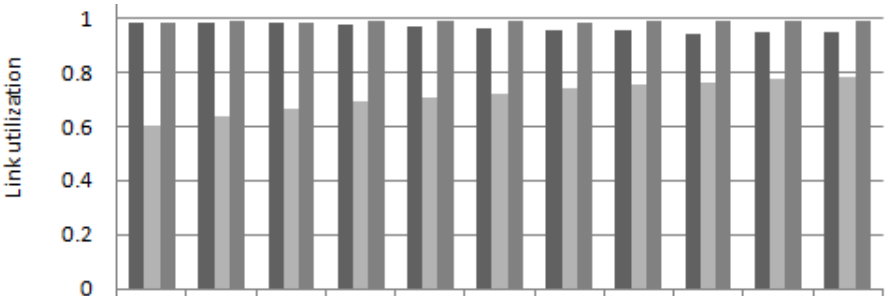


FIGURE 5.17. Average delay for RED, FRED and WTRED.

5.6.3. *Issues with RED's and FRED's implementations*

Erroneous parameter configuration degrades the performance of RED [64]. FRED proposed a new parameter conguration for RED in an effort to increase network performance. Unfortunately, FRED has numerous drawbacks.

FRED uses a very high max_p value. In some phases this value is ten times the initial value in RED. Given the same queue conditions, sometimes FRED will drop ten times as many packets as RED. The maximum threshold in



	1	2	3	4	5	6	7	8	9	10	11
RED	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.95	0.94	0.95	0.94
FRED	0.60	0.64	0.66	0.69	0.70	0.72	0.73	0.75	0.76	0.77	0.78
WTRED	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.98	0.99

FIGURE 5.18. Link utilization for RED, FRED and WTRED.

this case is actually reduced, resulting in a lower average queue size and lower average delay. Although FRED lowers delays, its overall performance is poor due to lower throughput, link utilization and higher loss rates, as demonstrated in section 5.6.2.

The suggested exponent value of the weight parameter using the normalized notation is -3. For example, the default value for w_q in RED is 0.002. When the actual queue size exceeds the warn threshold, FRED starts to increase the weight parameter. In Table 5.2, sub-phase 6, FRED multiplies the weight parameter by 20. In this case, w_q is not just doubled, it is also shifted one decimal point.

TD does not maintain an EWMA. RED maintains the average between the minimum and maximum thresholds while allowing transient bursts of traffic [29]. Higher weights mean that the average queue size will closely follow the actual queue size. In case of bursty traffic overwhelming the gateway for an extended period, FRED will behave like a TD algorithm.

5.7. Summary

Random Early Detection (RED) uses parameters, such as: maximum threshold, minimum threshold, weight and maximum drop probability to control congestion in TCP networks. Some RED variants use different parameter

settings to enhance RED's implementation. FRED dynamically adjusts the maximum drop probability and the weight parameter to increase network performance. This chapter highlights some of RED's and FRED's drawbacks and proposes the Weighted RED (WTRED) strategy to overcome these shortcomings.

WTRED uses a dynamic weight parameter and new maximum threshold and minimum threshold. NS-2 simulations show that FRED reduces loss rates and delays; but this comes at the cost of dramatically reduced throughput and link utilization. Comparatively, RED generates higher throughput and higher link utilization with acceptable delays and losses. WTRED provides the highest throughput and link utilization. The packet loss rate is slightly higher than RED, but the benefit of these slightly higher losses is a lower average delay. Overall, the results suggest that WTRED provides a better balance of throughput, loss, delay and network utilization.

CHAPTER 6

Conclusions

Queue management is an approach used to control congestion in TCP networks. Traditional queue management strategies, including Tail Drop (TD), used a First In First Out (FIFO) technique to control congestion. These techniques drop packets from the intermediate router queue depending on a predefined threshold or drop level. Whenever the actual queue size exceeds the prefixed threshold; TD drops packets from the tail of the queue. TD and traditional queue management strategies had served the network for an extended period of time until the detection of congestion collapse [55] [54].

TD and traditional queue managent strategies are reactive and commonly referred by the Passive Queue Management (PQM). Lock out, full queue and global synchronization are the most severe problems incurred by using the TD strategy.

The subsequent congestion control strategies used a proactive approaches to avoid congestion before buffers overflow. Hence, these approaches are reffered by the Active Queue Management (AQM). Early Random Drop (ERD) is one of the earliest implementations of the AQM. ERD proposes a threshold less than the buffer limit. Whenever the actual queue size exceeds the threshold, ERD drops packets randomly from the queue using a prefixed probability.

Due to the use of fixed drop probability and threshold; ERD is considered to have bias against bursty traffic and causes global synchronization problems. Random Early Detection (RED) was proposed with two thresholds and a dynamic drop probability. Instead of using the actual queue size check; RED used a low pass filter to calculate the average queue size. Thus, RED can more effectively avoid global synchronization and bias against bursty traffic.

Many RED-based strategies have been proposed to enhance the performance, however, RED and its variants have some disdvantagies. The mismatch between the average and actual queue sizes degrades the network performance. In addition, parameter configuration in RED is very sensitive. Erroneous configuration makes RED behave like TD. This thesis invistegated the mismatch and parameter configuration problems and proposed solutions for RED's drawbacks.

6.1. Proposed solutions

Traditional congestion control strategies; such as TD, used the actual queue size check to determine congestion. Instead of the actual queue size check, RED used the weight parameter to calculate the average queue size avg in order to determine congestion. Both approaches have disadvantages. All TD problems are caused by the actual queue size check; such as global synchronization and lock out. The average queue size check disregards the actual peaks in the queue; maintaining low avg. This mismatch between the average and actual queue sizes leads to buffer overflow for long period of time. In this situation, RED behaves same as TD. Thus, the risk of full queue and global synchronization phenomena is high.

Some RED-based strategies proposed dynamic parameter configuration to overcome RED's disadvantages. Adaptive RED (ARED) proposed a technique to prevent buffer overflow by adjusting the maximum drop probability max_p dynamically. Other strategies; such as Effective RED (ERED), used both actual and average queue sizes to control congestion. Refined RED (FRED) used dynamic weight (w_q) and maximum drop probability (max_p) to enhance the performance of RED.

This thesis proposed three RED-based strategies to ovecome the mismatch between the average and actual queue sizes. New parameter configurations are proposed for each strategy to avoid the drawbacks of erroneous configuration. The proposed strategies are: Queue Sectors RED (QSRED),

Risk Threshold RED (RTRED) and Weighted RED (WTRED). These strategies are simulated using the NS-2 simulation and their performance is evaluated and compared with current RED-based strategies.

6.2. Simulation results

QSRED divides the buffer into six equal sectors. These sectors represent new dropping levels added to the original RED implementation. QSRED used the actual and average queue size parameters to help RED absorb short lived bursty traffic and control TCP congestion efficiently. Since RED uses probabilistic packet dropping, QSRED dynamically adjusts the max_p value of RED to maintain network stability and smooth traffic. QSRED is compared with ERED and RED. The results show that QSRED offers higher throughput and link utilization with lower packet losses and delays.

RTRED is a compromise between the TD and RED strategies. RTRED uses both the actual and average queue sizes to calculate the drop probability and congestion level. These calculations operate in conjunction with a third drop level; the risk threshold. The simulation results suggest that RTRED outperformed competing RED-based strategies; reducing the unnecessary packet losses, the average delay time and the overhead of packet retransmission. Furthermore, RTRED avoids wasting the gateway buffer and increases the buffer utilization.

WTRED uses dynamic weight parameter and new maximum threshold and minimum threshold. NS-2 simulations show that, a similar strategy, FRED reduces loss rates and delays; but this comes at the cost of dramatically reduced throughput and link utilization. Comparatively, RED generates higher throughput and higher link utilization with acceptable delays and losses. WTRED provides the highest throughput and link utilization. The packet loss rate is slightly higher than RED, but the benefit of these slightly higher losses is a lower average delay. Overall, the results suggest that WTRED provides a better balance of throughput, loss, delay and network utilization.

6.3. Future work

Future work will further investigate the optimal usage of the weight parameter. Additionally, in the future we plan to compare the performance of QSRED, RTRED and WTRED.

6.3.1. *Optimal weight parameter*

The weight parameter w_q is most critical in RED for two reasons. Firstly, large weights w_q prevent RED from filtering out transient congestion [29]. Secondly, low w_q prevents RED from detecting the intial stages of congestion. The main goal of RED is to maintain a desirable average queue zise which is less than a certain threshold. Equation 2.6 [29] illustrates this scenario; where L is the number of packet arrivals provided that the queue was empty initially, w_q is the weight parameter and min_{th} is the minimum threshold.

Theoretically, RED should maintain an average queue size less than the minimum threshold with few fluctuations less than the maximum threshold to absourb transient congestion. Practically, the average queue size in RED does not reflect the actual dynamics of the queue size and packets are dropped due to buffer overflow. This has been shown through simulation in this thesis. As a future work, we suggest including the queue idle and peak times in estimating the weight parameter dynamically. The main issue is to find the optimal w_q that reasonably reflects the dynamics of the actual queue size over the average queue size.

6.3.2. *Comparison between QSRED, RTRED and WTRED*

Using the actual queue size check as the only congestion indicator in TD causes problems; such as global synchronization, lock out and full queue. QSRED and RTRED apply actual and average queue size checks to match the queue dynamics. In addition to the complexity incurred by the double queue size check, QSRED and RTRED add more sectors and drop levels which

means more monitoring and more parameter configuration. Actual queue size dynamics in WTRED are reflected on the average queue size using adjustable weight parameter w_q. WTRED also maintains the original drop levels of RED which makes WTRED the simplest strategy to implement.

QSRED and RTRED use the maximum drop probability max_p to adjust the drop rate depending on the level of congestion. The congestion level in both strategies is partially estimated using the actual queue size. WTRED assigns a specific weight parameter w_q for every drop level to increase/decrease the drop rate. The congestion level in WTRED is estimated depending on the average queue size. Hence, QSRED and RTRED are expected to be better than WTRED in solving the short lived congestion. Conversely, WTRED is expected to better solve the long lived congestion.

A comparative study between QSRED, RTRED and WTRED is a worthwhile point of research. As a future work, the performance of these three strategies can be evaluated against throughput, link utilization, packet loss and delays. The strategies can be simulated using the NS-2 simulator.

References

[1] B. Abbasov and S. Korukouglu. Effective red: An algorithm to improve red's performance by reducing packet loss rate. *Journal of Network and Computer Applications*, 32(3):703–709, May 2009.

[2] M. Allman, V. Paxon, and W. Stevens. Tcp congestion control. Technical report, IETF RFC2581, 1999.

[3] J. Aweya, M. Ouellette, and D. Montuno. A control theoretic approach to active queue management. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 36(2-3):203–235, July 2001.

[4] R. Axelrod. The evolution of cooperation. *HarperCollins*, 1984.

[5] D. Bansal and H. Balakrishnan. Bionomial congestion control al-gorithms. In *Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings*, volume 2, pages 631–640, Anchorage, AK, 2001.

[6] I. Barrera, G. Arce, and S. Bohacek. Statistical approach for congestion control in gateway routers. *Computer Networks*, 55(3):572–582, 2011.

[7] B. Braden, D. Clark, J. Crowcroft, and et al. Recommendations on queue management and congestion avoidance in the internet. Technical report, IETF RFC2309, April 1998.

[8] L. S. Brakmo and L. L. Peterson. Tcp vegas: End-to-end con-gestion avoidance on a global internet. *IEEE JSAC*, 13(8):1465–1480, 1995.

[9] H. Chengchen and L. Bin. Red with optimized dynamic threshold deployment on shered buffer. In *Advanced Information Networking and Applications AINA, 18th International Conference*, pages 451 – 454, 2004.

[10] M. Christiansen, K. Jeffay, D. Ott, and F. Smith. Tuning red for web traffic. *IEEE/ACM Transactions Networking*, 9(3):249–264, Jun 2001.

[11] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[12] V. Dumas, F. Guillemin, and P. Robert. A markovian analysis of additive-increase multiplicative-decrease algorithms. *JSTOR*, 34(1):85–111, March 2002.

[13] K. Fall and S. Floyd. Simulation-based comparisons of tahoe, reno, and sack tcp. Technical report, ACM Computer Communication Review, Berkeley, 1996.

[14] L. Feng, Z. Guan, and H. Wang. Controlling bifurcations and chaos in tcpudpred. *Real World Applications*, 11(3):1491–1501, 2010.

[15] W. Feng. A self-configuring red gateway. In *Proceedings of IEEE INFOCOM 99*, pages 1320–1328, San Franscisco, CA, March 1999.

[16] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A new class of active queue management algorithms. Technical report, University of Michigan, UM CSE-TR-387-. 99, April 1999.

[17] W. Feng, D. Kandlur, D. Saha, and K. G. Shin. Techniques for eliminating packet loss in congested tcp/ip networks. Technical report, University of Michigan, UM CSE-TR-349-97, November 97.

[18] S. Floyd. Tcp and explicit congestion notification. *ACM. Computer Communication Review*, 24(5):10–23, 1994.

[19] S. Floyd. *Notes on testing RED implementation, http://www.icir.org/floyd/papers/redtesting*, 1996.

[20] S. Floyd. Congestion control principles. *IETF*, September 2000.

[21] S. Floyd. Recommendation on using the gentle-variant of red, available: http://icir.org/floyd/red/gentle.html. 2000.

[22] S. Floyd. *References on RED (Random Early Detection) Queue Management: http://www.icir.org/floyd/red.html*, 2008.

[23] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Trans. Net.*, 1999.

[24] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *Networking, IEEE/ACM Transactions*, 7(4):458 – 472, 1999.

[25] S. Floyd, R. Gummadi, and S. Shenker. *Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, http://www.aciri.org/floyd/papers/adaptiveRed.ps*, 2001.

[26] S. Floyd and T. Hennderson. The newreno modification to tcp's fast recovery algorithm. Technical report, RFC 2582, 1999.

[27] S. Floyd and V. Jacobson. Traffic phase effects in packet-switched gateways. *ACM Comp. Commun. Rev.*, 21(2):26–42, April 1991.

[28] S. Floyd and V. Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience*, 3(3):115–156, September 1992.

[29] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1:397–413, 1993.

[30] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. *IEEE/ACM Transactions on Networking (TON)*, 2(2):122 – 136, 1994.

[31] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti. Congestion control mechanisms and the best effort service model. *Network, IEEE*, 15(3):16–26, May 2001.

[32] L. Guo and I. Matta. The war between mice and elephants. In *In Proceedings of the 9th IEEE International Conference on Network Protocols (ICNP)*, USA, November 2001.

[33] E. S. Hashem. Analysis of random drop for gateway congestion control. Master's thesis, Massachusetts Institute of Technology, 1989.

[34] M. Hassan and R. Jain. *High Performance TCP/IP Networking: Concepts, Issues, and Solutions.* Prentice-Hall, 2003.

[35] E. Head and S. Head. A survey on congestion control. *Global Journal of Computer Science and Technology*, 9(5):82–87, 2010.

[36] J. Hoe. Improving the start-up behavior of a congestion control scheme for tcp. Technical report, Proceedings of the ACM SIGCOMM, 1996.

[37] ISI. The network simulator - ns-2. *www.isi.edu/nsnam/ns-build.html*, 2005.

[38] V. Jacobson. Congestion avoidance and control. *Proc. ACM SIGCOMM 88*, 18(4):314–329, August 1988.

[39] J. M. Jaffe. Bottleneck flow control. *IEEE Trans. Commun*, 29(7):954–962, 1981.

[40] R. Jain and K. K. Ramakrishnan. Congestion avoidance in computer networks with a connectionless network layer: Concepts, goals and methodology. In *in Proc. IEEE Comp. Networking Symp, Washington D.C.*, pages 134–143, 1988.

[41] J. Kay and J. Pasquale. The importance of non-data touching processing overheads in tcp/ip. *in Proceeeding of the SIGCOMM Symposium on communications Architectures and protocols*, 23(4):259–268, 1993.

[42] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (avq) algorithm for active queue management. In *Proc. of ACM SIGCOMM 2001*, pages 123–134, San Diego, USA, august 2001.

[43] L. Le, J. Aikat, K. Jeffay, and F. Smith. The effect of active queue management on web performance. In *SIGCOMM'03*, pages 265–276, Karlsruhe, Germany, 2003.

[44] D. Lin and R. Morris. Dynamics of random early detection. *ACM SIGCOMM '97*, 27(4):127–137, September 1997.

[45] D. Lin and R. Morris. Dynamics of random early detection. In *ACM SIGCOMM '97*, pages 27–137, Cannes, France, 1997.

[46] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999.

[47] J. K. MacKie-Mason and H. R. Varian. Pricing congestible network resources. *IEEE Journal in Selected Areas*, 13(7):1141–1149, 1995.

[48] M. Mathis and J. Mahdavi. Refining tcp congestion control. In *Forward acknowledgment*, pages 281–292. in Procedings of the ACM SIGCOMM, 1996.

[49] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. Tcp selective acknowledgement options. Technical report, RFC 2018, 1996.

[50] M. May, C. Diot, B. Lyles, and J. Bolot. Influence of active queue management parameters on aggregate traffic performance. Technical Report RR3995, INRIA, 2000.

[51] M. May and et al. Reasons not to deploy red. pages 260–262, London , UK 1999, 1999.

[52] V. Misra, W. Gong, and D. Towsley. Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. *ACM SIGCOMM Computer Communication Review*, 30(4):151–160, October 2000.

[53] D. Mitra and J. Seery. Dynamic adaptive windows for high speed data networks with multiple paths and propagation delays. In *Proceedings. IEEE INFOCOM '91. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s.*, volume 1, pages 39–48, Bal Harbour, FL, Apr 1991.

[54] J. Nagle. Congestion control in ip/tcp internetworks. *RFC 896, IETF*, 1984.

[55] J. Nagle. On packet switches with infinite storage. *IEEE Trans. Commun.*, 35:435–438, 1987.

[56] T. J. Ott, T. V. Lakshman, and L. H. Wong. Sred: Stabilized red. *INFOCOM '99*, 3:1346–1355, 1999.

[57] J. Padhye, V. Firoiu, D. Towsley, and J. F. Kurose. Modeling tcp reno performance: a simple model and its empirical validation. *Networking, IEEE/ACM Transactions*, 8(2):133 – 145, 2000.

[58] P. K. Panos Gevros, Jon Crowcroft and S. Bhatti. Congestion control mechanisms and the best effort service model. *IEEE Network*, 15:16–26, 2001.

[59] M. Parris, K. Jeffay, and F. Smith. Lightweight active router-queue management for multimedia networking. In *Multimedia Computing and Networking, SPIE Proceedings Series*, volume 3020, San Jose, CA, January 1999.

[60] J. Postel. Transmission control protocol. *IETF*, RFC 793, 1981.

[61] W. Prue and J. Postel. Something a host could do with source quench. Technical Report RFC1016, IETF, July 1987.

[62] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ip. Technical report, IETF RFC2481, 1990.

[63] V. Rosolen, O. Bonaventure, and G. Leduc. A red discard strategy for atm networks and its performance evaluation with tcp/ip traffic. *ACM SIGCOMM Computer Communication Review*, 29(3):23–43, July 1999.

[64] S. Ryu, C. Rump, and Q. Chunming. Advances in internet congestion control. *Communications Surveys & Tutorials, IEEE*, 5(1):28–39, 2003.

[65] T. Sheldon and B. Sur. Congestion control mechanisms, 2001.

[66] S. Shenker, L. Zhang, and D. Clark. Some observations on the dynamics of a congestion control algorithm. *ACM Computer Communication Review*, pages 30–39, 1990.

[67] W. Stevens. Tcp slow-start, congestion avoidance, fast retransmit and fast recovery algorithms. Technical report, IETF RFC2001, 1997.

[68] L. Sun and L. Wang. A novel red scheme with preferential dynamic threshold deployment. *the International Conference on Computational Intelligence Security Workshops*, 2007.

[69] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choundhury. Design considerations for supporting tcp with per-flow queuing. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 299 –306, San Francisco, CA,

1998.

[70] C. Villamizar and C. Song. High performance tcp in ansnet. *ACM SIGCOMM Computer Communication Review*, 24(5):45–60, October 1994.

[71] L. Vukadinovic and L. Trajkovic. Red with dynamic thresholds for improved fairness. In *ACM Symposium on Applied Computing*, 2004.

[72] H. Wang and K. Shin. Refiend design of early detection gateways. *Global Telecommunication Conference GLOBCOM'99*, 1:769–775, 1999.

[73] L. Wei-yan, Z. Xue-lan, L. Tao, and L. Bin. A dynamic and self-adaptive tcp friendly congestion control mechanism in next-generation networks. In *Intelligent Systems and Applications ISA, Inter-national Workshop*, pages 1–4, Wuhan, 2009.

[74] M. Welzl and W. Eddy. Congestion control in the rfc series. *IRTF, RFC-5783*, 2010.

[75] C. Wu and S. Yang. The mechanism of adapting red parameters to tcp traffic. *Computer Communications*, 32(13-14):1525–1530, 2009.

[76] L. Y. Jie and Z. L. Bf-red: A novel algorithm for improving band-width fairness of red. In *The 2006 IEEE International Conference ICNSC*, volume 1001-1005, Ft. Lauderdale, FL, 2006.

[77] P. Young. *Recursive estimation and time-series analysis.* Springer-Verlag, 1984.

[78] L. Zhang. A new architecture for packet switching network protocols. Technical report, Laboratory for Computer Scienc, Massachusetts Institute of Technology, 1989.

[79] L. Zhang and D. Clark. Oscillating behavior of network traffic: A case study simulation. *Internetworking Research and Experience*, 1(2):101–112, 1990.

[80] L. Zhang, S. Shenker, and D. Clark. Observations on the dynamics of a con-gestion control algorithm: The effects of two way traffic. In *ACM SIGCOMM'91*, pages 133–148, Zurich, Switzerland, 1991.

[81] Y. Zhang and L. Qiu. Understanding the end-to-end perfor-mance impact of red in a heterogeneous environment. Technical report, Cornell CS Technical Report 2000-1802, July 2000.