

Modified Random Early Detection (RED) Technique Using Various Congestion Indicators

التعديل على تقنية الكشف العشوائي المبكر بإستخدام مؤشرات التدفق المختلفة

Student

Mohammad Ramez Abbas Ali

Supervisor Dr. Ahmad Adel Abu-Shareha

Thesis Submitted in Partial Fulfilment of the Requirements of the Degree of Master of Computer Information Systems

Department of Computer Information Systems

Faculty of Information Technology

Middle East University

January - 2017

Authorization

I, Mohammad Ramez Abbass Ali authorize Middle East University (MEU) to provide libraries, organization and even individuals with copies of my thesis when required.

Name: Mohammad Ramez Abbass Ali

Date: 7/2/2017Signature:

Examination Committee Decision

This is to certify that the thesis entitled "Modified Random Early Detection (RED) Technique Using Various Congestion Indicators" was successfully defended and approved on 18/1/2017.

Examination Committee Member

Signature

1. PhD Ahmad K. A. Kayed



2. PhD Mohammad MustfaAlhajHusan



3. Dr Ahmad Adel. Abu-shareha



Acknowledgments

In the first, my thanks are hereby extended to my God, then to Dr. Ahmad Adel Abu-shareha for his supportive and helpful supervision, as well as for assisting a student in every step of the project, and for providing important information and basics, which was very important for the successful implementation of the project. Further thanks are extended to everyone who helped me develop my understanding of the various nuances of the project and for everyone who believes that the knowledge is right for everyone

Dedication

I would like to exploit this opportunity to dedicate this project to my father, mother, my wifes, my Daughters, sun, brothers and sisters, without whose invaluable support. I would have ot been able to have achieved this in my lifetime.

May God bless them

TABLE OF CONTENTS

1
II
Ш
IV
V
VI
VIII
Х
XI
XII
XIV
1
3
5
6
7
7
7
8
9
9
9
14
17
17
21
28
29
29
30

3.2.1. Queue Length	31
3.2.2. Load Rate	31
3.2.3. Delay	33
3.3. The Proposed Sub-Methods	34
3.3.1. Sub-Method 1	35
3.3.2. Sub-Method 2	36
3.4. Summary	48
Chapter Four: THE EXPERIMENTAL RESULTS	49
4.1 Introduction	49
4.2 Environment	50
4.3 Experimental Setup	51
4.4 Results	53
4.4.1 Queue-based Proposed Methods	54
4.4.2 Delay-based Proposed Methods	59
4.4.3 Load-based Proposed Methods	64
4.4.4 Best of the Proposed Methods	70
4.5 Summary	75
Chapter Five : CONCLUSIONS and FUTURE WORKS	76
5.1 Conclusion	76
5.2 Future Work	77
References	79

LIST OF FIGUERS

Figure 1.1 Data Transmission Intermediated by Routers	6
Figure 1.2 The Router Location between Networks	2
Figure 1.3 The Router Buffer	3
Figure 1.4 The packets dropping	4
Figure 1.5 Flow chart for Random Early Detection	6
Figure.2.1 Buffer Queue and RED Parameters	13
Figure 2.2 Flowchart of Blue Algorithm	19
Figure. 2.3 WRED drops packets probability	24
Figure 2.4 FRED processing arriving packet	25
Figure 2.5 Flow chart for Basic CHOKe	27
Figure 4.1 Flowchart of the Experiments	53
Figure 4.2 Delay Comparison for the Proposed Queue-based Methods	55
Figure 4.3 Drop Comparison for the Proposed Queue-based Methods	56
Figure 4.4 Loss Comparison for the Proposed Queue-based Methods	57
Figure 4.5 Drop and Loss Comparison for the Proposed Queue-based Methods	59
Figure 4.6 Delay Comparison for the Proposed Delay-based Methods	60
Figure 4.7 Drop Comparison for the Proposed Delay-based Methods	61
Figure 4.8 Loss Comparison for the Proposed Delay-based Methods	62
Figure 4.9 Drop and Loss Comparison for the Proposed Delay-based Methods	64
Figure 4.10 Delay Comparison for the Proposed Load-based Methods	65

Figure 4.11 Drop Comparison for the Proposed Load-based Methods	66
Figure 4.12 Loss Comparison for the Proposed Load-based Methods	68
Figure 4.13 Drop and Loss Comparison for the Proposed Load-based Methods	69
Figure 4.14 Delay Comparison for the Best of the Proposed Methods	71
Comparison for the Best of the Proposed Methods	72
Figure 4.16 Loss Comparison for the Best of the Proposed Methods	73
Figure 4.17 Drop and Loss Comparison for the Best of the Proposed Methods	74

LIST OF TABLES

Table 2.1: comparison of mechanisms indicator	28
Table 3.1 The Modified-RED Sub-Methods	34
Table 4.1 Parameter settings	51
Table 4.2 Delay Comparison for the Proposed Queue-based Methods	54
Table 4.3 Drop Comparison for the Proposed Queue-based Methods	56
Table 4.4 Loss Comparison for the Proposed Queue-based Methods	67
Table 4.5 Drop and Loss Comparison for the Proposed Queue-based Methods	58
Table 4.6 Delay Comparison for the Proposed Delay-based Methods	60
Table 4.7 Drop Comparison for the Proposed Delay-based Methods	61
Table 4.8 Loss Comparison for the Proposed Delay-based Methods	62
Table 4.9 Drop and Loss Comparison for the Proposed Delay-based Methods	63
Table 4.10 Delay Comparison for the Proposed Load-based Methods	65
Table 4.11 Drop Comparison for the Proposed Load-based Methods	66
Table 4.12 Loss Comparison for the Proposed Load-based Methods	67
Table 4.13 Drop and Loss Comparison for the Proposed Load-based Methods	69
Table 4.14 Delay Comparison for the Best of the Proposed Methods	70
Table 4.1 Drop Comparison for the Best of the Proposed Methods	72
Table 4.16 Loss Comparison for the Best of the Proposed Methods	73
Table 4.17 Drop and Loss Comparison for the Best of the Proposed Methods	74

List of Abbreviations

<u>Abbreviations</u> <u>Meaning</u>

RED Random Early Detection

ERED Effective Random Early Detection

SFB Stochastic Fair Blue

GRED Gentle Random Early Detection

WRED weighted Random Early Detection

RRED Robust Random Early Detection

FRED Flow Random Early Detection

TCP Transmission Control Protocol

FDPS Flow-state-dependent dynamic priority

AVG Average queue size

QOS Quality of service

DP Dropping Probability

Modified Random Early Detection (RED) Technique Using Various Congestion Indicators

Prepared By

Mohammad Ramez Abbass Ali

Supervised By

Dr. Ahmad Adel Abu-Shareha

Abstract

In this thesis, modified Random Early Detection (RED) algorithm is proposed by including various selected congestion indicators. First, the best congestion indicators to be used for queue management are chosen. Then, these indicators, Queue length, load rate and Delay, are integrated with RED algorithm. Subsequently, nine different proposed methods were developed.

The proposed approach, as similar to the existing AQM method, preserves the core of the RED technique embodied in calculating Dp with each arrival packet, drop packets based on the calculated Dp and divide Dp calculation into categories. Existing AQM methods have taken different approaches in modifying RED. However, the overall trends in these approaches are changing both, the congestion indicator and the utilized Dp calculation procedure, building on the assumption that different indicators required different calculation procedure. Unlike the existing AQM methods, the proposed approach changes the RED's utilized congestion indicator and preserves the RED calculation. The proposed approach uses novel indicators in the RED framework. These indicators will be discussed accordingly.

The evaluation and comparison of the proposed methods shows that the proposed methods gain the best delay and best loss. However, ERED provides the best results to the dropping values. Subsequently, each of the proposed methods can be used according to the type of the network.

Keywords: Congestion control, Random Early Detection, Average queue size, Dropping Probability, Traffic control, Active Queue Management.

التعديل على تقنية الكشف العشوائي المبكر بإستخدام مؤشرات التدفق المختلفة

اعداد محمد رامز عباس علي اشراف الدكتور احمد عادل ابو شربحه

الملخص

في هذه الرسالة , لقد اقترح تعديل خوارزمية (RED) من خلال اختيار مؤشرات التدفق أو الاختتاق . اولا , لقد استخدم أفضل مؤشر تدفق تم اختياره لإدارة قائمة الانتظار .وبالتالي هذه المؤشرات , وطول قائمة الانتظار , ومعدل التحميل والتأخير . تتكامل مع خوارزمية (RED) . وفي وقت لاحق سيتم وضع تسعة طرق مختلفة.

النهج المقترح، مشابهة لأسلوب AQM موجود، يحافظ على جوهر تقنية (RED)المجسدة في حساب Dp مع كل حزمة الوصول، إسقاط الحزم على أساس حساب Dp وتقسيم العملية الحسابية Dp إلى فئات. ان طرق AQM الموجوده اتخذت نهجاً مختلفة في تعديل (RED).

ومع ذلك، تغيير الاتجاهات العامة في هذه النهج متغير لكلاهما,ان مؤشر الازدحام وإجراءات حساب Dp المستخدمة، بناء على افتراض أن مختلف المؤشرات تتطلب إجراء حساب مختلفة على حد سواء،. خلافا لأساليب AQM موجودة، يتغير مؤشر الازدحام المستخدمة في (RED) للنهج المقترح ويحافظ على حساب (RED). حيث ان النهج المقترح يستخدم مؤشرات جديدة في إطار (RED). وستناقش هذه المؤشرات وفقا لذلك.

ويبين تقييم ومقارنة الأساليب المقترحة أن الأساليب المقترحة تكسب أفضل تأخير وافضل خسارة. ومع ذلك، يوفر ERED أفضل النتائج لإسقاط القيم. بعد ذلك، يمكن استخدام كل من الأساليب المقترحة وفقا لنوع الشبكة.

كلمات المفتاحية : التحكم في الازدحام , الكشف العشوائي المبكر , متوسط حجم قائمة الانتظار , احتمال الاسقاط , التحكم بالإختناق , إدارة قائمة الانتظار النشطة .

CHAPTER ONE

INTRODUCTION

Internet is enormous small networks that are linked together to form the global network for human being over this planet. Over the years, Internet has become an essential part of human needs, as more and more people are surfing the Internet continuously as part of their daily lives. Through the Internet, people are reading electronic news, searching information, watching videos, playing online games, and talk to each other via p2p telephone services. The traditional circuit-switched telephone networks are now evolving into packet-switched networks. This is because packet switched networks can provide extra and a variety of communication services as well as reducing the cost of running and maintaining these services.

Data communication through the global network, or what so called the Internet, is transmitted from a source device to a destination and passing by cables, routers and other intermediate devices and carrier medium, as illustrated in Fig 1.1. These devices, which form the communication medium, consist of a set of hardware (physical equipment) and software (programs). Router is one of the most important, yet hardly handled device in the communication channels.

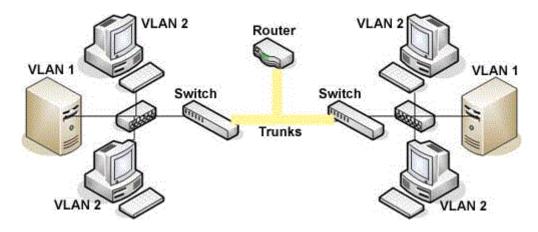


Fig 1.1: Data Transmission Intermediated by Routers (Dobbins,1998)

The router, as illustrated in Fig 1.2, is connected to at least two networks, commonly two LANs or WANs or a LAN and it is responsible for directing the data between the connected networks. Router mission is to keep the data flowing between networks and maintain the networks connectivity with the global network. The routers making the Internet work by re-directing data based on a uniform addressing scheme. Information could be sent to anywhere in the world as long as the site has an IP address (Pirenne, 2015).

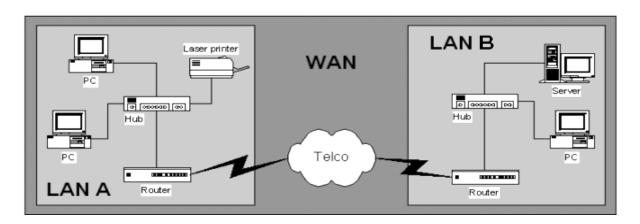


Fig 1.2: The Router Location between Networks (Lammle,2013)

1.1. The Router Buffer and The Congestion Problem

All Internet routers contain storage space to hold packets that are arrived to the router. Arrival packets are accommodated in the router buffer to be processed and then transmitted into their destination, as illustrated in Fig. 1.3 (Spalink et al.,2001). The storage space also absorbing bursty traffic to avoid loss of packets. However, it leads to puts delay on the transmitted packets, which raises an important question about the optimal size of these storage. Notably, small storage leads to the data loss, as the transmitted packets in bursty traffic leads to over flow the routers quickly and all the following packets will be forced to be lost as it cannot be entered to the router. Packet loss affects the performance of applications badly. On the other hand, the large router storage increase latency, complexity and cost required.

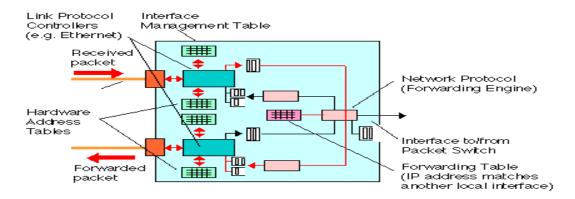


Fig. 1.3: The Router Buffer (Spalink, 2001)

Congestion is a problem that occurs on shared networks when many users try to gain access to the same resources (bandwidth, stores, and queues). For example, congestion at the highway where many cars continuously intervened regardless of existing of high traffic. With the entry of more cars on the highway, the congestion increased and leads to bad consequences in the end, such as ramps back up, preventing vehicles from getting at all. Congestion at the router occurs when the number of arrived

packets exceeded the capacity of the router buffer and eventually leads to buffer overflow and packet loss (Baklizi et al., 2014).

Congestion control techniques and mechanisms at the router can prevent congestion before it happens, or remove the tension, after it happened. Overall congestion control techniques can be divided into two categories, one category prevents congestion from happening while other category removes congestion as it occurred. The first one is using open loop control traffic, and the second uses a closed loop congestion control in an attempt to remove the congestion after it occurs (Lim, 2015).

The congestion prevention technique implements packets dropping, as illustrated in Fig. 1.4, when the number of packets in the router storage reachs a specific critical limit prevent packet loss. Low packet dropping in critical cases may lead to packet loss and high drop in non-critical case degrade the network performance.

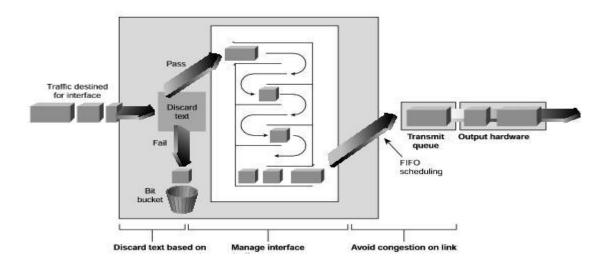


Fig. 1.4: The Packets Dropping (Chua,2007)

Subsequently, the goal of the congestion control mechanisms is to achieve the best packet dropping rate to avoid packet loss without degrading the performance of the network by adding more delay.

1.2. Random Early Detection (RED)

Random early detection (RED) algorithm, which was proposed by Floyd and Jacobson in 1993 (Floyd et al., 1993) was designed with the objectives to minimize packet loss and queuing delay, avoid global synchronization of sources, maintain high link utilization and remove biases against bursty sources. The basic idea behind RED queue management is to detect congestion early and to convey congestion notification to the end-hosts, allowing them to reduce their transmission rates before queues in the network overflow (Feng et al., 2002).

Random early detection algorithm (RED) was recommended later in the IETF. The RED's goal is to avoid global synchronization of flows in TCP and then maintain high productivity. RED has been suggested to reduce delays and achieve a fair distribution despite the number of connections in TCP (Floyd et al., 1993).

RED computes a weighted average queue length in a router to determine when congestion is occurring. When the average queue length is below minth (minimum threshold), no packets are marked. While, when the average is between minth (minimum threshold) and maxth (maximum threshold), RED marks incoming packets with probability p (where p varies linearly between 0 and maxp). When the average is above maxth, all incoming packets are dropped (p=1) as illustrated in Fig 1.1.(Balkas et al., 2002).

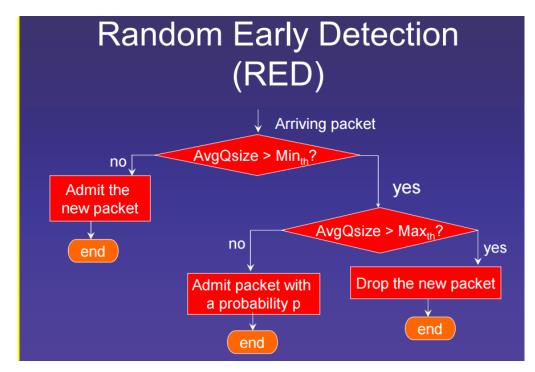


Fig 1.1: Flow chart for Random Early Detection (Misra,2010)

RED uses the average queue length as a congestion indicator, which forms a critical part of the RED algorithm. While, several other algorithms were proposed, such as Weighted Fair Queue (Homg et al.,2001), RED remains the most utilized and well-known method for its simplicity, consistency and acceptable performance (Rosolen et al.,1999). However, the congestion indicator of the RED's algorithm was not criticized.

1.3. Problem statement:

The major problem is to investigate how the modify on RED algorithm through congestion indicators are adjusted so that the flow of the algorithm RED indicator while maintaining the key equation where you must determine the best probability of dropping depending on different situations at the router buffer so that it can be a no-congestion, pre-congestion, light-congestion and heavy congestion. The researcher also chooses congestion indicators, which will be replaced by the previous studies are then replaced the indicator's

original algorithm and then work experiences and evaluation of this algorithm, after the amendment to the indicators and the preservation of the main equation.

1.4. Problem Statement Questions:

- How to determine and criticize of best dropping probability with different situations at the router buffer, which are: no-congestion, pre-congestion, lightcongestion and heavy congestion?
- How to choose the best congestion indicators to be used for queue management?
- How to modify RED algorithm by including the selected congestion indicators?
- How to implement, test and evaluates the modified RED algorithm?

1.5. Objective:

The main objective of this research is to define the best dropping probability while having a different with situations at the router buffer, which are: no-congestion, precongestion, light-congestion and heavy congestion, and select the best congestion indicators, by reviewing and criticizing the previous work in the field of active queue management. Then modify RED algorithm by modifying both, the main dropping calculation in RED and the RED's dropping categories, and implement, test and evaluates the modified RED algorithm.

1.6. Motivation:

In this study, several methods proposal in order to reduce and prevent the congestion of data through the modification to the congestion indicators of RED algorithm, which is still so far used in some organizations

while maintaining the original equation indicators and thus can use these methods suggested in the different types of networks to prevent the congestion of data before they occur thus reducing delays in data and improve data network.

1.7. Methodology:

The adopted methodology approach in this research is experimental, which involved the following main steps:

We must define the best dropping probability while having a different with situations at the router buffer, which are: no-congestion, pre-congestion, light-congestion and heavy congestion.

- 1. Then select the best congestion indicators, by reviewing and criticizing the previous work in the field of active queue management.
- 2. After that modify RED algorithm by replace indicator with another indicator, the main dropping calculation in RED and the RED's dropping categories.
- 3. Last thing make implement, test and evaluates the modified RED algorithm.

.

Chapter Two

Background and Related Work

2.1. Introduction

As mentioned earlier, congestion is a problem that occurs on shared networks when many users try to gain access to the same resources (bandwidth, stores, and queues). Congestion control techniques and mechanisms can prevent congestion before it happens, or remove the tension, after that it happened. This chapter gives a brief review on these mechanisms with the focus on the Active Queue Management (AQM) methods.

2.2. Active Queue Management and RED

There are mainly two ways to deal with congestion: Active and Passive. The early method for congestion control was passively act after congestion occur with the aim to reduce the bad consequences that results from the congestion occurrences. Active Queue Management (AQM) is a term that are given for the congestion control methods that manage, detect and prevent congestion actively. These set of methods use a set of indicators to predict and prevent congestion in the early stage (De Vos,2012).

Random Early Detection (RED) algorithm was proposed by Floyd and Jacobson in 1993 as the first Active Queue Management (AQM) mechanism, which was, later on, standardized as a recommendation in the IETF. The goal of RED was to avoid global synchronization of TCP flows and maintain high productivity. Moreover, RED was proposed to reduce delay and achieve fair allocation though multiple TCP connections (Li, 2008).

RED calculates the average queue size using a low pass filter with an exponential weighted moving average. The calculated average queue size is compared with two thresholds: a minimum and a maximum threshold.

When the average queue size is less than the minimum threshold, no packets are marked, because this is indicating that the buffer is of fair size and no congestion would occur at this stage. While, when the average queue size is greater than the maximum threshold, every arriving packet is marked and dropped. This is because at this stage, the buffer is about to be overflowed by the influence of a congestion state and packets are about to be lost. Thus, dropping packets will reduce the number of packets in the buffer and prevent buffer overflowing. When the average queue size is between the minimum and maximum thresholds, each arriving packet is marked and dropped with probability Dp, where Dp is a function of the average queue size (avg). The probability that a packet is marked from a particular connection is roughly proportional to that connection's share of the bandwidth at the router.

RED algorithm implements its process in two stages: One is for computing the average queue size, which determines the degree of burstiness in the router buffer. It takes into account the period when the queue is empty (the idle period) by estimating the number m of small packets that could have been transmitted by the router during the idle period and the number of packets resides in the buffer over a period of time. RED algorithm is given in Algorithm 2.1.

Algorithm 2.1: RED

```
INITIALIZATION:
2
        avg := 0
        count := -1
       FOR EACH arrival packet
       CALCULATE new avg as follows:
         IF q==0 THEN avg:=(1-w)^{f(time-q\_time)} * avg
6
         IF q != 0 THEN avg := (1-w)^* avg + w_q *q
       CALCULATE Dp and its related parameters, and implements packet dropping, as:
       if (\min_{th} \le avg < \max_{th})
10
        increase count
11
        Dp' = D_{\text{max}} * (avg - \min_{\text{th}}) / (\max_{\text{th}} - \min_{\text{th}})
12
         Dp = Dp'/(1-\text{count}*Dp') + \text{w}_{d}(D)
13
        with probability Dp
14
         drop and mark packet
15
         count := 0
16
        else if (avg > max_{th})
17
        drop and mark packet
       Count = 0
18
19
       else
20
        Count = -1
21
        When q==0
22
       q_time=time
```

12

Saved Variables:

avg: average queue size

q_time: start of the queue idle time

count: packets since last marked packet

Pre-Initialized Parameters:

w_q: queue weight

minth: minimum threshold for queue

max_{th}: maximum threshold for queue

D_{max}: maximum value for Dp

Other:

Dp: current packet-marking probability

Time: current time

As avg varies from min_{th} to max_{th} , the packet-marking probability Dp varies linearly from 0 to D_{max} . The final packet-marking probability Dp increases slowly as the count increases since the last marked packet, to prevent, to some extent, consequent dropping of packets. Fig. 2.1 illustrates a queue buffer with RED supported queue management.

The problem with RED is the pre-initialized parameters, max_{th}, min_{th}, D_{max} and w_q that should be given a certain value in-order to give a satisfactory QoS.

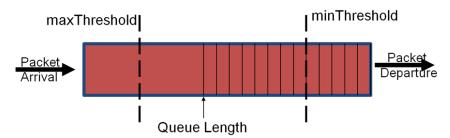


Fig.2.1: Buffer Queue and RED Parameters (Bonald,2000)

Active Queue Management (AQM) methods have been developed to monitor, detect and prevent congestion in early stage. These methods have been designed to keep queue at the router buffer as small as possible and to provide early notification of congestion. The main technique of these methods is to drop packets when necessary inorder to prevent router overflowing. At the same time, this technique avoids dropping packets unnecessarily (Mohamed et al., 2010).

Active Queue Management (AQM) methods help TCP to carry out links utilization properly. Active queue management determines routers quantitative and qualitative packets dropping. Subsequently, AQM methods reduce the number of packets loss in routers. By keeping the average queue size small, AQM methods provide greater capacity to absorb naturally occurring bursts without dropping packets and provide lower-delay interactive service by keeping the average queue size small, queue management will reduce the delays seen by flows (Baker et al., 2015).

Among many methods, the most prominent method is the random early detection (RED), which was proposed in the early nineties. RED controls the queue length so it is used in a lot of routers webserver (Xie et al., 2008).

2.3. Related Work

This section reviews the proposed techniques for AQM, such as FRED (Silberschatz et al.,2006), BLUE (Dhodapkar et al.,2002), SFB (Thiruchelvi et al. 2008), and CHOKe (Thiruchelvi et al. 2008). A survey on active queue management mechanisms. International Journal of Computer Science and Network Security, 8(12), 130-145. and compares some of these techniques with RED Lee et al., (2008) and Drop Tail (Floyd et al.,2000), which are considered the base line for congestion control. The review and comparison focus on the utilized indicators, performance and simulation results. The characteristics of these techniques are also discussed and compared.

In the efforts to achieve high Quality of Service (QoS), many several congestion control approaches were developed. Floyd and Jacobson (1993) proposed Random Early Detection (RED) approach with the aim at detecting and preventing congestion in the packet-switched. Gateway detects congestion status with the reference to the average size of the queue in the buffer. The average number of packets in the queue determins whether to drop or mark packets by placing a bit in the header of the packet to notify the sender about congestion.

When the average queue size exceeds a predetermined threshold, RED drops or marks every arriving packet with a certain probability, where the probability is a function of the average size of the queue. Subsequently, Red maintains a fair average size of the queue while allowing Bursts every now. RED was designed to accompany congestion control with no bias against irregular senders and to avoid global synchronization of the many connections decreasing their window at the same time. Overall, RED is relatively simple and easy to be implemented in an existing networks or with a newly established high-speed networks.

Since RED was proposed, many AQM methods and algorithms were proposed with reference to RED and with the aim to overcome some of the expected limitations in its procedure.

Feng et al. (2002), proposed BLUE algorithm with the aim to address the problem of solely depends on the average queue size as congestion indicator to calculate Dp, as given in RED. Blue uses history of packet loss to manage congestion in the buffer. In addition to BLUE, SFB, a novel algorithm for scalable flows in a large aggregate was proposed. Using SFB, all the connected flows are denied from exceeding its limited rate and by increasing Dp for such flows. Using both simulation and controlled trials, BLUE was proved to give better performance compared to RED, both in terms of packet loss and the size of the over time.

Siew (2005), proposed Flow-state-dependent dynamic priority scheduling (FDPS), a new mechanism that depends on building a scheduling and traffic monitoring. FDPS depends on the queue size as indicator of the buffer states. The proposed mechanism drops packets of specific source when the source exceed the limits allocated and there is no space to accommodate the exceeded packets. The results show that FDPS can differentiating services and prevent congestion.

Lee et al. (2008) proposed congestion control based on servo control structure based on Linear Quadratic, servo-LQ. The implemented approach uses the traditional control mechanism with an input variable that represents the queue size. The simulation results have shown that the proposed controller gives satisfactory performance balance between the queue size and packet loss.

Abbasov et al., (2009) proposed an extension to RED algorithm, called HERED. The proposed algorithm used a dropping function that is similar to the one used in RED, but it drops less packet when the load is light and more aggressive dropping, compared to RED, is implemented when the load is heavy. The simulation shows that it achieves better QoS compared to RED and the state-of-art algorithms.

Chen et al., (2010) proposed RED- restraint algorithm, which aims to keep the queue around a stable target value. The RED-restraint algorithm stabilize the queue size by adjusting the dropping probability value as the value of the current queue length shifted away from the target value. RED-restraint differs from RED by using the actual queue length instead of the average queue length in RED. Moreover, RED-restraint stabilize q around target value, which is not presented in RED. The simulations show that RED-restraint gives better results compared to RED in packet loss and dropping rate.

As experimental studies, (Wang, Y. C. et al 2004) conducts a statistical analysis of the behaviour of RED. According to the experiments, RED shows weakness dealing with heavy congestion, where many packets are lost due to the slow response of the avg. When the number of packets arrived to the queue increased with the status of heavy congestion, the calculated (avg) increases slowly, because it depends strongly on the previous value of avg besides the new value of the queue size. Subsequently, avg takes time to cope with the increase of packets resides in the buffer.

Ren et al., (2011) conducted a comparative study of the different congestion control schemes based on some key performance metrics. By comparing different algorithms, it was proven experimentally that there is no mechanism that can efficiently control congestion and all of the congestion control mechanisms required large number of parameters tuning, which affect the system. In addition, it also concludes that in today's

high-speed network, and the nature of congestion is not really known, which suggest to use different types of congestion control.

Subsequently, there is a need to propose new congestion control that does not depend on manual tuning of parameters. Moreover, there is a need to have a congestion control that does not concern about pre-assumptions about the nature of congestion that faces the network. The utilization of machine learning can solve both of the problems if congestion control problem can be reformulated in away to suits the available machine learning algorithms.

2.3.1. Average-Queue based Methods

RED (Wang, B. et al 2005) uses the average queue length as the indicator to estimate the state of the buffer and decide about the dropping probability. RED was the first technique in AQM and many other methods has follow the same concepts in buffer management and congestion control. When the link is congested, RED randomly drops arriving packets even if they would fit into the queue, to signalize congestion to the end nodes. The probability of the packet dropping is a function of the average queue length, while RED is adequate in situations with moderate congestion levels, it has been shown, that – depending on its parameters – the queue length either oscillates, or the technique reacts to the changes in traffic very slowly (Brazio et al., 2006).

2.3.2. Packet-Loss based Methods

BLUE (Dhodapkar et al.,2002) uses packet loss as congestion indicator. BLUE is one of the newly proposed techniques for congestion control – either using ECN-marking or packet dropping.

If the queue losses packets due to queue overflows, the probability is increased. If the link is underutilized, the probability is decreased. To avoid oscillations, it freezes the probability after every change for a fixed time interval, Note that RED cannot achieve this if the queue length is oscillating.

Using both simulation and experimentation, BLUE is shown to overcome many of RED's shortcomings. RED has been designed with the objective to (1) minimize packet loss and queuing delay, (2) avoid global synchronization of sources, (3) maintain high link utilization, and (4) remove biases against bursty sources, and BLUE either improves or matches RED's performance in all of these aspects (Dhodapkar et al., 2002).

The key idea behind BLUE is to perform queue management based directly on packet loss and link utilization rather than on the instantaneous or average queue lengths. This is in contrast to other active queue management schemes which use some form of queue occupancy in their congestion management.

BLUE maintains a single marking probability, which it uses to mark or drop packets. If the queue is continually dropping packets due to buffer overflow, BLUE increments marking probability, thus increasing the rate at which it sends back congestion notification. Conversely, if the queue becomes empty or if the link is idle, BLUE decreases its marking probability. This effectively allows BLUE to (learn) the correct rate it needs to send back congestion notification. The BLUE technique, which the marking probability is updated when the queue length exceeds a certain value as illustrated in Fig 2.2.(Feng et al., 2006).

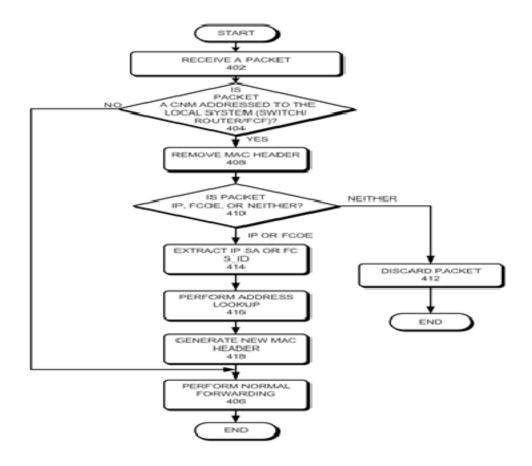


Fig 2.2: Flowchart of Blue Algorithm (Feng, 2006)

This modification allows room to be left in the queue for transient bursts and allows the queue to control queuing delay when the size of the queue being used is large. Besides the marking probability, BLUE uses two other parameters which control how quickly the marking probability changes over time. The first is freeze time, which determines the minimum time interval between two successive updates of marking probability. This allows the changes in the marking probability to take effect before the value is updated again. Freeze time is initialized as a constant, this value should be randomized in order to avoid global synchronization. The other parameters used, (d1 and d2), determine the amount by which marking probability is increased when the queue overflows or is decreased when the link is idle. The parameter d1 is set significantly larger

than d2, this is because link underutilization can occur when congestion management is either too conservative or too aggressive, but packet loss occurs only when congestion management is too conservative. By weighting heavily against packet loss, BLUE can quickly react to a substantial increase in traffic load. Note that there are a myriad of ways in which marking probability can be managed, experiments with additional parameter settings and algorithm variations have also been performed with the only difference being how quickly the queue management algorithm adapts to the offered load. While BLUE seems extremely simple, it provides a significant performance improvement even when compared to a RED queue which has been reasonably configured (Feng et al., 2006).

Another technique that uses packet loss as congestion indicator is Stochastic Fair Blue (SFB) (Thiruchelvi et al. 2008). SFB is a novel technique for protecting TCP flows against non-responsive flows, which was built based on BLUE. SFB is a FIFO queuing algorithm that identifies and rate-limits non-responsive flows based on accounting mechanisms similar to those used with BLUE. SFB maintains accounting bins that are organized in L levels with N bins in each level. In addition, SFB maintains L independent hash functions, each associated with one level of the accounting bins. Each hash function maps a flow into one of the accounting bins in that level. The accounting bins are used to keep track of queue occupancy statistics of packets belonging to a particular bin. As a packet arrives at the queue, it is hashed into one of the N bins in each of the L levels. If the number of packets mapped to a bin goes above a certain threshold (i.e., the size of the bin), the packet dropping probability marking probability for that bin is increased. If the number of packets in that bin drops to zero, marking probability is decreased. The observation is that, a non-responsive flow quickly drives marking probability to 1 in all of the L bins it is hashed into. Responsive flows may share one or two bins with nonresponsive flows, however, unless the number of non-responsive flows is extremely large compared to the number of bins, a responsive flow is likely to be hashed into at least one bin that is not polluted with non-responsive flows and thus has a normal value. The decision to mark a packet is based on P_{min} the minimum marking probability value of all bins to which the flow is mapped into. If P_{min} is 1, the packet is identified as belonging to a non-responsive flow and is then rate-limited.

2.3.3. Queue and Average-Queue based Methods

Gentle Random Early Detection (GRED) (Sally et al., 2000), was proposed in order to increase throughput and reduce the undesired oscillation in buffer size of router by enhancing parameter settings of RED. GRED was evaluated using same simulation as it used in RED.

GRED aims to solve some of RED's problems using technique that is similar to RED, but the main difference is in parameter setting in order to be optimized and have a better performance regarding to Packet loss and throughput. In GRED another parameter was introduced namely, Effective Random Early Detection (ERED) (Freed et al., 2006) was proposed to reduce packet loss rates in a simple and scalable manner. ERED modifies the packet drop function of RED scheme by controlling packet dropping function both with average queue size and instantaneous queue size. Simulations demonstrate that ERED achieves a highest throughput and lowest packet drops than RED and it performs better than RED due to lowest packet drops.

RED probability is dropped by a mechanism dependent on queue length of buffer and TCP senders are informed before congestion. The mechanism monitors average queue length at a router and a drop probability is calculated accordingly, if the average length of waiting increased, congestion will happen and therefore the dropping probability should also increase to prevent congestion (Wang, B. et al 2005).

At light traffic load when the average queue size exceeds the maximum threshold (max_{th}), RED drops all packets even though current queue size is small or queue is empty. When the load is getting heavy and the current queue size quickly approaches the queue limit—an indicator that the queue size may soon get out of control, but the average queue size is not big enough to make random drops; ERED allows more aggressive packet dropping to quickly back off from it.

The disadvantage of RED is if congestion is sufficiently heavy that the gateway cannot control the average queue size. ERED proposed to control average queue size when connections immediately reduce their sending rate in the case of no congestion (Janevski et al.,2003).

When the average queue size is between the minimum and the maximum threshold, each arriving packet is dropped with probability, even though current queue size is small or queue is empty. ERED proposed to calculate packet dropping probability according to instantaneous queue size when queue size increases immediately and exceeds queue limit, but average queue size is below the minimum in the case of congestion, and drop each arriving packet with probability (Xu et al.,2005).

ERED has higher throughput and lower packet loss rate than other AQM algorithms. ERED has highest throughput value between simulated algorithms when comparing throughput values of AQM algorithms and the Router buffer of ERED algorithms newer overflows and reduces to zero during simulation. Router buffer frequently reduces to zero in RED because RED is aggressive when traffic load is light and not aggressive when traffic load is heavy. ERED forwarded the most packets to destination nodes and has lost the least packets among rest techniques (Abbasov et al.,2009).

Weighted random early detection (WRED) (May et al.,1999) is a congestion avoidance mechanism and drops them when queues are full, WRED depends on the value of precedence in measuring the size of the waiting lists and starts to drop packets when the wait between the minimum and maximum threshold list and arranging will decide that 1 in every N packets are dropped. WRED helps to prevent TCP synchronization and TCP starvation but when TCP loses packets it will go into slow start and if all TCP sessions lose packets at the same time they could become synchronized (Wurtzler et al.,2002).

Random early detection (RED) is a mechanism to avoid congestion which takes advantage of the control mechanism in the congestion of the TCP. By randomly dropping packets before high congestion occurs, RED reduces the source packet transmission rate, WRED drops packets on a selective basis on the IP precedence. Edge routers set the IP precedence to packets as they enter the network. WRED is useful on any output interface where you expect it to be crowded. However, WRED is usually used in the core routers for the network, and not on the edge. WRED uses these precedence to determine how to deal with different traffic (Odom et al., 2004).

When a packet arrives, the average is less than the minimum queue threshold, the arriving packet is queued. But if the average is between the minimum queue threshold for that type of traffic and the maximum threshold for the interface, the packet is either dropped or queued, depending on the packet drop probability for that type of traffic. But if the average queue size is greater than the maximum threshold, the packet is dropped as illustrated in Fig 2.3.

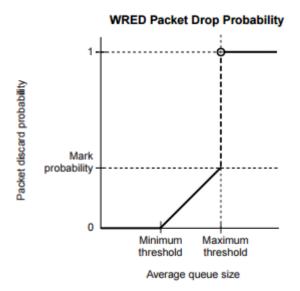


Fig. 2.3: WRED drops packets probability (May,1999)

The packet drop probability is based on the minimum threshold and maximum threshold, and mark probability denominator. But when the average queue depth is upper the minimum threshold then RED starts dropping packets. The average of packet drop increases linearly as the average queue size increases until the average queue size reaches the maximum threshold. The mark probability denominator is the part of packets dropped when the average queue depth is at the maximum threshold, and when the average queue size is upper the maximum threshold, all packets are dropped (Wurtzler et al.,2002).

Robust random early detection (RRED) (Zhang et al., 2010) is a queuing correction for a network scheduler. RRED technique was suggest to improve the TCP throughput against LDoS attacks. The main idea behind RRED is to detect and filter out attack packets before a normal RED algorithm is applied to incoming flows. RRED algorithm can significantly improve the performance of TCP under Low-avrage negation-of-service attacks , and the basic idea behind the RRED is to detect and filter out LDoS attack packets from incoming flows before they feed to the RED algorithm (Braden et al.,1998) .

Flow Random Early Drop (FRED) (Silberschatz et al., 2006) aims to reduce the effects of injustice in RED. Instead of indicating congestion to randomly selected contacts through a drop packets relatively speaking, that generates a unique responses of the selective action of a group that has filtered of connections that have a large number of packets in the queue. FRED is able to isolate the non-passage of greed adapt more effectively (Pan et al., 2000).

FRED is like RED, but with some additions. FRED introduces the parameters minq and maxq, they aims for the minimum and maximum number of packets each flow should be allowed to buffer, and FRED maintains a variable strike for each flow, which counts the number of times the flow has failed to respond to congestion notification. FRED held to account for the presence of flows with high values of strike (Kidambi et al.,2000).

FRED is modified version of the RED, providing selective dropping on the basis of an active share of the flow of the charges buffer. FRED keeps this only extra state for flows that have packets stored in each gateway (Alemu et al.,2004), which is compatible with existing FIFO queue architectures.. FRED processes arriving packets using the following flow chart of the algorithm illustrated in Fig 2.4.

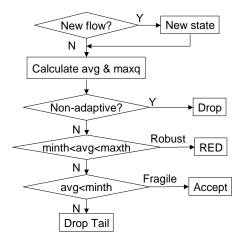


Fig 2.4: FRED processing arriving packet (Stoica, 1998)

CHOKe as a queue management algorithm, CHOKe basicly will detect all the non-responsive and unwanted flows using pre-existing queue buffer occupancy information of each flow. The RED usually used a certain existing technique in order to calculate the average occupancy and it works, and that was the same used by CHOKe (Bergmeyer et al., 2012).

It also marks two thresholds on the buffer, a minimum threshold minth and a maximum threshold maxth. Depending on the Queue size the outcome will change: If the average queue size is less than minth, each arriving packet is being automatically queued and waited into the FIFO buffer. If the collected arrival rate is less than the output link capacity, the average queue size should not build up to minth very often and packets are not dropped frequently. If the average queue size is more than maxth, each arriving packet is dropped (Pan, R., Prabhakar, B., 2000).

This will take the queue occupancy back to below maxth. When the average queue size is bigger than minth, each arriving packet is compared with a another packet this is done randomly, named as drop candidate packet, from the FIFO buffer. If they have the same flow ID, they are both dropped. Otherwise, the randomly chosen packet is kept in the buffer (in the same position as before) and the arriving packet is dropped with a probability that depends on the average queue size. The drop probability is computed exactly as in RED. In particular, this means that packets are dropped with probability 1 if they arrive when the average queue size exceeds maxth. And in order to bring the queue occupancy back to below maxth as fast as possible. A flow chart of the algorithm is given in Fig 2.5 (Pan et al., 2000).

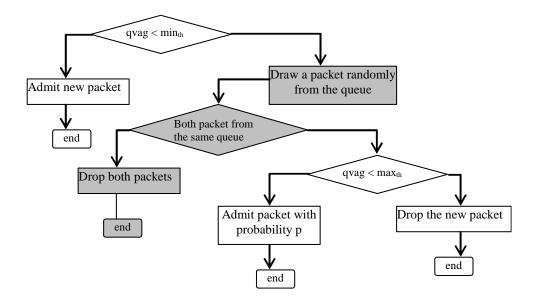


Fig 2.5: Flow chart for Basic CHOKe (Pan,2000)

The following table 2.1 shows the comparison brief review on these mechanisms indicator:

Table 2.1: comparison of mechanisms indicator

	Indicator	Avoid Global Synchronization	Predict Congestion in Early stage	Response to sudden	Avoid unnecessary	No delay	Avoid Parameterization
RED	avg	Yes	Yes	No	Yes	No	No
FRED	avg	Yes	Yes	No	Yes	No	No
RRED	avg & q	Yes	Yes	No	Yes	No	No
WRED	avg & q	Yes	Yes	No	Yes	No	No
GRED	avg & q	Yes	Yes	No	Yes	No	No
СНОКе	avg	Yes	Yes	Yes	No	No	No
ERED	avg & q	Yes	Yes	Yes	Yes	No	No
BLUE	PL	Yes	No	No	Yes	Yes	No
SFP	PL	Yes	No	No	Yes	Yes	No
AVQ	Arrival rate	Yes	Yes	Yes	No	Yes	No
FDPS	q	Yes	Yes	Yes	No	No	No
AFRED	avg	avg	Yes	Yes	No	Yes	Yes

2.4. Summary

Several queue management algorithms (RED, FRED, BLUE, SFB, CHOKe) based on comparison result and algorithm characteristics. It's still hard to conclude which algorithm is better in all aspects than another, especially considering the deployment complexity. But the major trends are: (1) all these algorithms have in common that they do provide high link utilization, (2) RED and BLUE do not usually identify and drop the non-responsive flow, but the other three algorithms FRED, SFB and CHOKe maintains equal sharing among different traffic flows, (3) the equality maintained by the three algorithms is achieved by using different methods, FRED record per-active-flow information, SFB statistically multiplex buffers to bins, but it requires to achieve this is to have a large number of non-responsive flows, CHOKe correlates dropping rate with corresponding flow's incoming rate, and is able to drop large number of non-responsive flows adaptively, (4) all of the algorithms has computation overhead per incoming packet, they do require a different space.

Chapter Three

Proposed Work

This Chapter presents the proposed approach for AQM. The proposed approach operates in the router buffer in order to control the number of packets presents in the router and drop the packets randomly before buffer overflow, it calculates a dropping probability with each arrival packet and it avoids the problem of global synchronization phenomena. Overall, the proposed approach contributes by using different indicators with RED, as will be discussed in this chapter.

This chapter is organized as follows: Section 3.1. is an introduction. Section 3.2 presents the proposed methods. Section 3.3. is the summary of the chapter 3.

3.1. Introduction

RED, the first and most well-known AQM methods is influenced by the utilized congestion indicator. The congestion indicator in RED, the average queue length, has two main roles, these are:

- Selecting *Dp* category.
- Calculating the actual *Dp* value in likelihood dropping category [0-1].

Subsequently, the proposed approach and the other existing AQM methods use different indicators to play these roles, accordingly.

The proposed approach, as similar to the existing AQM method, preserves the core of the RED technique embodied in calculating Dp with each arrival packet, drop

packets based on the calculated Dp and divide Dp calculation into categories. Existing AQM methods have taken different approaches in modifying RED. However, the overall trends in these approaches are changing both, the congestion indicator and the utilized Dp calculation procedure, building on the assumption that different indicators required different calculation procedures. Unlike the existing AQM methods, the proposed approach changes the RED's utilized congestion indicator and preserves the RED calculation. The proposed approach uses novel indicators in the RED framework, these indicators will be discussed accordingly.

3.2. The proposed methods

RED uses Average Queue Length as single and sole congestion indicator.

Average Queue Length is an intelligent indicator of the number of packets within the buffer zone. It is a reflection of the queue length, an indicator of the actual number of packets in the router buffer.

Different indicators rather than AVG-Queue are investigated. The utilized indicators are calculated in a way to fit in the RED procedure. Subsequently, different calculation process than those founds in the literature is implemented. The investigated indicators are: Queue Length, Load Rate and Delay.

As noted and proven in the experiments, we found out that whenever the number of arrival queue increase, the indicators will increase their number .

The indicators were used in order to replace the parameters in the equation of original RED in order to have different proposed methods than the original RED .

3.2.1. Queue Length

Queue length is the number of the packets that reside in the router buffer simultaneously at a specific time. It is simply calculated by counting the number of packets that reside in the buffer. Queue length was previously used in an extended-RED AQM method, called ERED. The utilization of this factor along with the average queue length, as given in ERED work, has shown an improvement in the overall results. However, in the work, Queue length is used as in different way that is simpler compared to ERED.

3.2.2. Load Rate

Load Rate is an indicator of the ratio between the packet arrival rate and packet departure rate that formulate the load on the router. The load rate is calculated based on four scenarios as given in Algorithm 1.

Algorithm 1: Lode Rate Calculation

- 1. if(QueueLength==0)
- 2. LoadRate = 0
- 3. else if (QueueLength==BufferSize)
- 4. LoadRate = 1
- 5. else if(DepartureRate>ArrivalRate)
- 6. LoadRate =0
- 7. else

8. LoadRate = ArraivalRate-DepartureRate

In Line 1 and Line 2, the load rate is set to zero if there is no packets queued in the buffer. In another case, in Line 3 and Line 4, if the buffer is full, by other means if the number of queued packets is equal to the buffer capacity, then the load rate is set to one, which is the maximum loade rate value. In such a case, the load cannot be determined precisely, as no arrival packets can occur in this case. However because buffer can only be overflowed with heavy load, it is given the maximum load rate value.

In Line 5 and Line 6, if the departure rate is greater than the arrival rate, no load will be added on the buffer because the departed packet are larger than those arriving. Thus, the load rate is set to zero. Finally, the major load rate calculation is implemented in Line 7 and Line 8. This case refers to arriving packets greater than departing; buffer is neither empty nor full. In such a case the load rate will be given a value in the range [0-1] and it will be calculated as the difference between the arrival rate and departure rate. The arrival rate is calculated as given in Equation 3.1.

ArraivalRate_t = $w * Arrival_t + (1-w) * ArraivalRate_{t-1} (3.1)$

where, $ArraivalRate_t$ is the arrival rate calculate at the current time denoted as t. $ArraivalRate_t$ is calculated as low filter pass of the average arrival packets at the current time, $Arrival_t$, and the old arrival rate calculated at time (t-1), $ArraivalRate_{t-1}$. This averaging process is similar to the way of calculating the average queue length proposed by Floyd. The advantage of such low pass filter as mentioned before is to avoid false calculation in burst traffic or short ideal link. The weight w is set to a value less than 0.5 to ensure the low pass filtering of the old and new values.

 $Arrival_t$, the arrival at time t is calculated as the inverse difference between the current time, which is the time of a newly arrived packet, and the time of the last arrived packet, as given in Equation 3.2. Note that $Time_{current}$ is a time of newly arrived packet implicitly understood as all calculations in any AQM method is implemented with packet arrivals only.

 $Arrival_t = 1/(Time_{current} - Time_{Previous})$ (3.2)

The departure rate is calculated differently from the arrival because the calculation is not implemented with each packet departure, thus, the AQM has no control on the packet departing process. The departure at time t, *DepartureRate_t*, is calculated as the difference between the arrival rate and the packet queuing rate, which is estimated as *AVG/BufferSize*, as given in Equation 3.3.

DepartureRate_t = arrivalRate - (AVG/BufferSize) (3.3)

Note that, the values of LoadRate, $ArraivalRate_t$ and $DepartureRate_t$ are updated with each packet arrival, by other means with each AQM triggered for action.

3.2.3. Delay

Delay is an indicator of the average time the packets will be waited in the queue. The delay is calculated based on Equation 3.4.

Delay = LoadRate * QueueLength (3.4)

Generally, these indicators are the most common indicators used by the existing active queue management methods. It is used with different calculations rather than given here.

3.3. The Proposed Sub-Methods

The proposed approach extends RED by maintaining the overall structure of the underlying process in RED while modifying the major indicators used to produce the final Dp value. In the proposed work, the process is divided into three stages: which are:

The investigated Modified-RED Sub-methods are listed in Table 3.1.

Table 3.1: The Modified-RED Sub-Methods

Investigated Trial #	For Categorization	For [0-1] Category Calculation	Comments
RED	AVG-Queue	AVG-Queue	Original RED
1	AVG-Queue	Queue-Length	Proposed Variation 1
2	Queue-Length	AVG-Queue	Proposed Variation 2
3	Queue-Length	Queue-Length	Proposed Variation 3
4	AVG-Queue	Load-Rate	Proposed Variation 4
5	Load-Rate	AVG-Queue	Proposed Variation 5
6	Load-Rate	Load-Rate	Proposed Variation 6
7	AVG-Queue	Delay	Proposed Variation 7
8	Delay	AVG-Queue	Proposed Variation 8
9	Delay	Delay	Proposed Variation 9

3.3.1. Sub-Method 1

The first variation uses the original RED indicator average queue length (AVG) for categorization and the queue length for the main calculation. The proposed variation is given in Algorithm 1.

Algorithm 2: Proposed Sub-Method-1

- 1. QueueAverage:= 0
- 2. count := -1
- 3. with packet arrival
- 4. If QueueLenth==0 THEN QueueAverage:=(1-w)f(time-q_time) * QueueAverage
- 5. If QueueLenth<> 0 THEN QueueAverage:= (1-w)* QueueAverage + w *q
- 6. If (minth≤ QueueAverage<maxth)
- 7. Count ++
- 8. Dp'= Dmax* (QueueLenth -minth)/(maxth-minth)
- 9. Dp = Dp'/(1-count*Dp')
- 10. with probability Dp
- 11. drop and mark packet
- 12. count = 0
- 13. Else If (QueueAverage>maxth)
- 14. drop and mark packet
- 15. count = 0
- 16. Else
- 17. count = -1
- 18. If QueueLenth ==0
- 19. q_time=time

In Line 1 and Line 2, the parameters required to run the algorithm, are setting up to their initial values. The queue management process start at Line 3. The average queue length is calculated according to one of three scenarios, similar to the scenarios given in the original RED, in Line 4 and Line 5. In Line 6 to Line 12, the first calculation category

is represented, the likelihood category. As given in Line 8, the calculation of the dropping probability has been modified to include the QueueLength parameter in the place of the AverageQueue, which was used in the original RED. The new calculation of Dp is given in Equation 3.5.

 $Dp'=Dmax*(QueueLenth - min_{th})/(max_{th}-min_{th}), Dp = Dp'/(1-count*Dp')$ (3.5)

In Line 13 to Line 15, the second calculation category is represented, which is the full dropping category. This scenario for calculating Dp value is followed when AverageQueue is above the value of max_{th} . In Line 16 and Line 17, the third calculation category is represented, which is zero dropping category. This scenario for calculating Dp value is followed when AverageQueue is below the value of min_{th} . Finally, the value of ideal time is updated if the queue is getting full. The ideal time is used to calculate the AverageQueue.

3.3.2. Sub-Method 2

The second variation uses the queue length for categorization and the original RED indicator, average queue length (AVG), for the main calculation. This variation is given in Algorithm 2.

Algorithm 3: Proposed Sub-Method-2

- 1. QueueAverage:= 0
- 2. count:= -1
- 3. with packet arrival
- 4. If QueueLenth==0 THEN QueueAverage:=(1-w)f(time- q_time) * QueueAverage
- 5. If QueueLenth<> 0 THEN QueueAverage:= (1-w)* QueueAverage + w *q
- 6. If (minth \le QueueLenth \le maxth)
- 7. Count ++
- 8. Dp'= Dmax* (QueueAverage -minth)/(maxth-minth)
- 9.Dp = Dp'/(1-count*Dp')
- 10. with probability Dp

- 11. drop and mark packet
- 12. count = 0
- 13. Else If (QueueLenth>maxth)
- 14. drop and mark packet
- 15. count = 0
- 16. Else
- 17. count = -1
- 18. If QueueLenth ==0
- 19. q_time=time

As before Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management. The average queue length is calculated, similar to the calculation in the original RED, in Line 4 and Line 5. Line 6 to Line 12 represent the first calculation category, the likelihood category. As given in Line 6, the triggering of the category depends on comparing QueueLength value to the thresholds, which replace the AverageQueue, which was used in the original RED. The calculation of the Dropping probability, in Line 7 and Line 8 is implemented as the original RED. Lines 13 to 15 represent the second calculation category, full dropping category, which is followed when QueueLength is above the value of maxth, unlike original RED which uses AverageQueue for this purpose. Line 16 and Line 17 represent the third calculation category, no dropping category, which is followed when AverageQueue is below the value of minth. Finally, the value of ideal time is updated if the queue is getting full. The ideal time is used to calculate the AverageQueue.

Proposed Variation 3:

The third variation uses the queue length for categorization and for the main calculation. This variation is given in Algorithm 3.

Algorithm 3: Proposed Variation 3 1. QueueAverage:= 0 2. count := -13. with packet arrival 4. If $(\min_{th} \leq \text{QueueLenth} < \max_{th})$ 5. Count ++ 6. Dp'= Dmax* (QueueLenth -min_{th})/(max_{th}-min_{th}) 7. Dp = Dp'/(1-count*Dp')8. with probability Dp 9. drop and mark packet 10. count = 011. Else If (QueueLenth>max_{th}) 12. drop and mark packet 13.count = 014. Else 15. count = -1

As before, Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management .Line 4 to Line 10 represent the first calculation category, the likelihood category. As given in Line 4, the triggering of the category depends on comparing QueueLength value to the thresholds, which replace the AverageQueue, which was used in the original RED. The calculation of the Dropping probability in Line 5 and Line 6 is implemented based on QueueLength. Line 11 to Line 13 represent the second calculation category, full dropping category, which is followed when QueueLength is above the value of max_{th}. Line 14 and Line 15 represent the third calculation category, no dropping category, which is followed when AverageQueue is below the value of min_{th}. As noted there is neither. AverageQueue nor ideal time calculation in this variation.

Proposed Variation 4:

The fourth variation uses the original RED indicator average queue length (AVG) for categorization and the load rate for the main calculation. The proposed variation is given in Algorithm 4.

Algorithm 4: Proposed Variation 4

- 1. QueueAverage:= 0
- 2. count := -1
- 3. with packet arrival
- 4. If QueueLenth==0 THEN QueueAverage:=(1-w)f(time-q_time) * QueueAverage
- 5. If QueueLenth<> 0 THEN QueueAverage:= (1-w)* QueueAverage + w *q
- 6. loadRate = LoadRateCalculation()
- 7. If $(min_{th} \le QueueAverage \le max_{th})$
- 8. Count ++
- 9. Dp'= Dmax* (loadRate -min_{th})/(max_{th}-min_{th})
- 10. Dp = Dp'/(1-count*Dp')
- 11. with probability Dp
- 12. drop and mark packet
- 13. count:= 0
- 14. Else If (QueueAverage>max_{th})
- 15. drop and mark packet
- 16. count = 0
- 17. Else
- 18. count = -1
- 19. If QueueLenth ==0
- 20. q_time=time

As before, Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management. The average queue length is calculated, similar to the calculation in the original RED, in Line 4 and Line 5. The load rate is calculated as given in Algorithm 1 in Line 6. Line 7 to Line 13 represent the first calculation category, the likelihood category. As given in Line 9, the calculation of the Dropping probability has been modified to include the LoadRate parameter in the place of the AverageQueue,

which was used in the original RED. The LoadRate is calculated as discussed in Algorithm 1. Line 14 to Line 16 represent the second calculation category, full dropping category, which is followed when AverageQueue is above the value of max_{th}. Line 17 and Line 18 represent the third calculation category, no dropping category, which is followed when AverageQueue is below the value of min_{th}. Finally, the value of ideal time is updated if the queue is getting full. The ideal time is used to calculate the AverageQueue.

Proposed Variation 5:

The fifth variation uses the LoadRate for categorization and the original RED indicator, average queue length (AVG), for the main calculation. This variation is given in Algorithm 5.

Algorithm 5: Proposed Variation 5

- 1. QueueAverage:= $\overline{0}$
- 2. count := -1
- 3. with packet arrival
- 4. If QueueLenth==0 THEN QueueAverage:=(1-w)f(time- q_time) * QueueAverage
- 5. If QueueLenth<> 0 THEN QueueAverage:= (1-w)* QueueAverage + w *q
- 6. loadRate = LoadRateCalculation()
- 7. If $(min_{th} \le loadRate \le max_{th})$
- 8. Count ++
- 9. Dp'= Dmax* (QueueAverage -min_{th})/(max_{th}-min_{th})
- 10. Dp = Dp'/(1-count*Dp')
- 11. with probability Dp
- 12. drop and mark packet
- 13. count = 0
- 14. Else If (LoadRate>max_{th})
- 15.drop and mark packet
- 16. count = 0
- 17. Else
- 18. count = -1
- 19. If QueueLenth ==0
- 20. q_time=time

As before Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management. The average queue length is calculated, similar to the calculation in the original RED, in Line 4 and Line 5. The load rate is calculated as given in Algorithm 1 in Line 6. Line 7 to Line 13 represent the first calculation category, the likelihood category. As given in Line 7, the triggering of the category depends on comparing LoadRate value to the thresholds, which replace the AverageQueue, which was used in the original RED. The calculation of the Dropping probability, in Line 8 and Line 9, is implemented as the original RED. Line 14 to Line 16 represent the second

calculation category, full dropping category, which is followed when LoadRate is above the value of max_{th}. Unlike original RED which uses AverageQueue for this purpose. Line 17 and Line 18 represent the third calculation category, no dropping category, which is followed when LoadRate is below the value of min_{th}. Finally, the value of ideal time is updated if the queue is getting full. The ideal time is used to calculate the AverageQueue.

Proposed Variation 6:

The sixth variation uses the load rate for categorization and for the main calculation. This variation is given in Algorithm 6.

Algorithm 6: Proposed Variation 6

- 1. QueueAverage:= 0
- 2. count := -1
- 3. with packet arrival
- 4. LoadRate = LoadRateCalculation()
- 5. If $(\min_{th} \leq loadRate < max_{th})$
- 6. Count ++
- 7. Dp'= Dmax* (LoadRate -min_{th})/(max_{th}-min_{th})
- 8. Dp = Dp'/(1-count*Dp')
- 9. with probability Dp
- 10. drop and mark packet
- 11. count:= 0
- 12. Else If (LoadRate>max_{th})
- 13. drop and mark packet
- 14. count = 0
- 15. Else
- 16. count = -1

As before Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management. The load rate is calculated as given in Algorithm 1 in Line 4. Line 5 to Line 11 represent the first calculation category, the likelihood category. As given in Line 5, the triggering of the category depends on comparing LoadRate value to the thresholds, which replace the AverageQueue, which was used in the original RED.

The calculation of the Dropping probability, in Line 6 and Line 7 is implemented based on LoadRate. Line 12 to Line 14 represent the second calculation category, full dropping category, which is followed when LoadRate is above the value of max_{th}. Line 15 and Line 16 represent the third calculation category, no dropping category, which is followed when LoadRate is below the value of min_{th}. As noted there is neither AverageQueue nor ideal time calculation in this variation.

Proposed Variation 7:

The seventh variation uses the original RED indicator average queue length (AVG) for categorization and the delay for the main calculation. The proposed variation is given in Algorithm 7.

Algorithm 7: Proposed Variation 7

- 1. QueueAverage:= 0
- 2. count := -1
- 3. with packet arrival
- 4. If QueueLenth==0 THEN QueueAverage:=(1-w)f(time- q_time) * QueueAverage
- 5. If QueueLenth<> 0 THEN QueueAverage:= (1-w)* QueueAverage + w *q
- 6. LoadRate = LoadRateCalculation() DelayRate = LoadRate*QueueLength
- 7. If $(min_{th} \le QueueAverage \le max_{th})$
- 8. Count ++
- 9. Dp'= Dmax* (DelayRate -min_{th})/(max_{th}-min_{th})
- 10. Dp = Dp'/(1-count*Dp')
- 11 with probability Dp
- 12. drop and mark packet
- 13. count := 0
- 14. Else If (QueueAverage>max_{th})
- 15. drop and mark packet
- 16. count = 0
- 17. Else
- 18.count = -1
- 19. If QueueLenth ==0
- 20. q_time=time

As before Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management. The average queue length is calculated, similar to the calculation in the original RED, in Line 4 and Line 5. The load rate and delay are calculated as given in Algorithm 1 and Equation 4 in Line 6. Line 7 to Line 13 represent the first calculation category, the likelihood category. As given in Line 9, the calculation of the dropping probability has been modified to include the DelayRate parameter in the place of the AverageQueue, which was used in the original RED. The DelayRate is calculated as discussed in Equation 4. Line 14 to Line 16 represent the second calculation category, full dropping category, which is followed when AverageQueue is above the value of max_{th}. Line 17 and Line 18 represent the third calculation category, no dropping category, which is followed when AverageQueue is below the value of min_{th}. Finally, the value of ideal time is updated if the queue is getting full. The ideal time is used to calculate the AverageQueue.

Proposed Variation 8:

The eighth variation uses the DelayRate for categorization and the original RED indicator, average queue length (AVG), for the main calculation. This variation is given in Algorithm 8.

Algorithm 8: Proposed Variation 8

- 1. QueueAverage:= 0
- 2. count:= -1
- 3. with packet arrival
- 4. If QueueLenth==0 THEN QueueAverage:=(1-w)f(time- q_time) * QueueAverage
- 5. If QueueLenth<> 0 THEN QueueAverage:= (1-w)* QueueAverage + w *q
- 6. LoadRate = LoadRateCalculation() DelayRate = LoadRate*QueueLength
- 7. If $(\min_{th} \leq DelayRate \leq \max_{th})$
- 8. Count ++
- 9. Dp'= Dmax* (QueueAverage -min_{th})/(max_{th}-min_{th})
- 10. Dp = Dp'/(1-count*Dp')
- 11. with probability Dp
- 12. drop and mark packet
- 13. count:= 0
- 14. Else If (DelayRate>max_{th})
- 15. drop and mark packet
- 16. count = 0
- 17. Else
- 18. count = -1
- 19. If QueueLenth ==0
- 20. q_time=time

As before Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management. The average queue length is calculated, similar to the calculation in the original RED, in Line 4 and Line 5. The delay is calculated in Line 6. Line 7 to Line 13 represent the first calculation category, the likelihood category. As given in Line 7, the triggering of the category depends on comparing DelayRate value to the thresholds, which replaces the AverageQueue, which was used in the original RED. The calculation of the Dropping probability, in Line 8 and Line 9 is implemented as the original RED. Line 14 to Line 16 represent the second calculation category, full dropping category, which is followed when DelayRate is above the value of max_{th}. Unlike original RED which uses AverageQueue for this purpose. Line 17 and Line 18 represent the third calculation category, no dropping category, which is followed when DelayRate is below the value of min_{th}. Finally, the value of ideal time is updated if the queue is getting full. The ideal time is used to calculate the AverageQueue.

Proposed Variation 9:

The ninth variation uses the delay for categorization and for the main calculation.

This variation is given in Algorithm 9.

Algorithm 9: Proposed Variation 9

- 1. QueueAverage:= 0
- 2. count := -1
- 3. with packet arrival
- 4. LoadRate = LoadRateCalculation() DelayRate = LoadRate*QueueLength
- 5. If $(\min_{th} \leq DelayRate \leq \max_{th})$
- 6. Count ++
- 7. Dp'= Dmax* (DelayRate -min_{th})/(max_{th}-min_{th})
- 8. Dp = Dp'/(1-count*Dp')
- 9. with probability Dp
- 10. drop and mark packet
- 11. count:= 0
- 12. Else If (DelayRate>max_{th})
- 13. drop and mark packet
- 14. count = 0
- 15. Else
- 16. count = -1

As before Line 1, Line 2 and Line 3 setting the parameters to their initial values and initiate the queue management. The delay is calculated in Line 4. Line 5 to Line 11 represent the first calculation category, the likelihood category. As given in Line 5, the triggering of the category depends on comparing DelayRate value to the thresholds, which replaces the AverageQueue, which was used in the original RED. The calculation of the dropping probability, in Line 6 and Line 7 is implemented based on DelayRate.

Line 12 to Line 14 represents the second calculation category, full dropping category, which is followed when DelayRate is above the value of max_{th}. Line 15 and Line 16 represent the third calculation category, no dropping category, which is followed when DelayRate is below the value of min_{th}. As noted there is neither Average Queue nor ideal time calculation in this variation.

3.4. Summary

We replaced the existing original parameter code RED with new indicators (Queue length, load rate and Delay). As every time we do these replacements, we have a proposed methods, that provided us with nine different proposed methods. Later on in chapter four all these proposed methods will be tested and studied in order to find out which one of the nine is going to be recognised as the best proposed methods.

CHAPTER FOUR

THE EXPERIMENTAL RESULTS

This chapter presents, compares and discusses the results of the proposed methods for congestion control that were built based on RED, as has been discussed in Chapter Three. The proposed and compared methods are developed, executed and compared in this chapter.

This chapter is organized as follows: Section 4.1 presents an introduction to this chapter. Section 4.2 describes all the aspects that are related to the environment, in which the experiments were conducted. Section 4.3 describes the parameters and settings that are used in the experiments. The results are given in Section 4.4. Finally, Section 4.5 gives a summary of the chapter.

4.1 Introduction

The procedure of conducting the experiments — consists of implementing the proposed and compared methods, testing, and comparing the results. The implementation is conducted using JAVA programming language. The code is developed in NetBeans 7.5 IDE. Multiple tests were conducted by changing the rate of incoming packets, which provides different results in order to get variety of tests. The results are evaluated according to the following measures: Delay, which is known as the average time that packets will spend waiting in the queue. Loss usually happens when one or more packets of data travelling across a computer network fail to be accommodated in the router buffer. Dropping rate is the rate of packets that dropped by the AQM to the total number of packets. Sum of dropping and losing reflects the total number of packets that were not accommodated in the router buffer, either by loss or by dropping. Drop and loss were

aggregated, as each of them alone might not express the effectiveness of the compared approach. This is because sometimes, drooping are necessary to avoid loss and in some other times it is not necessary. In order to achieve a proper evaluation with the proposed methods, these methods are compared with RED, which is the core of the developed methods and ERED which is a unique AQM method that was proposed to overcome the limitations of RED.

4.2 Environment

The simulation of the network process is implemented using one of the well-known approaches called discrete time queue (Alfa, 2010). The Discrete Time Queue tracks measures and evaluates the status of the network and network resources at specific time intervals known as SLOTS. At every slot, either a packet arrive event or departed event separately or both events may occur at the same time. Where two subsequent packets arrival without departure makes two time slots and so on. Several methods have been introduced and tested using discrete-time queues. The other approach, called continuous model, measures and evaluates the network performance periodically with equal length periods. However, this approach does not properly address the events of packet arrival and departures accurately. As the AQM is based on calculating Dp with each arrival packets (event based), Discrete time queue was chosen to verify the proposed work.

The discrete time queue, packet arrival and packet departure rates are established as probability values. Probability for arrival, and similarly for departure, with 0 values, means no packets will be arrived at any time slot. While, with probability of 1.0, a packet will surely arrive at each time slot. For a value of 0.5, there is Probability for the packet to arrive or not at each time slot.

4.3 Experimental Setup

The probability of the packet departure in the conducted test is set to 0.5. The probabilities of packet arrival are set to the values of 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 0.99. When the arrival probability is below the departure, no congestion is expected. While, congestion or pre-congestion are expected when the arrival rate is higher than the departure rate.

Two Million time slots were used in the experiments. This value allows sufficient results. Part of these slots is used as a warm-up before the steady state, with total number of 800000 slots, and the rest is for the experimental measures. A buffer size of 20 packets was used (Baklizi, et 57 al., 2014). The parameters minth, maxth, maxp, and weight values are set to 3, 9, 0.1, and 0.002, respectively, as recommended in RED (Floyd and Jacobson, 1993). The parameters that are used in the experiments for the proposed and compared method, given in Table 4.1, these parameters are as follows:

Table 4.1: Parameter settings

Parameter	Values		
Packet Arrival Probability	0.3-0.99		
Packet Departure Probability	0.5		
Number of Slots	2,000,000		
Number of Slots for Warm-Up	800,000		
Router Buffer Capacity	20		
Queue Weight	0.002		
max _p	0.1		
\min_{th}	3		
max _{th}	9		

The experiments are conducted as follows: The parameters are initialized first to values that are given in Table 4.1. Then, a packet is generated and sent to the queue, based on the probability of packet arrival. If a packet is generated and sent to the router, there is a probability for the packet to be lost if the queue is full, or it might be dropped or queued as decided based on the calculated value for the drop probability Dp. DP is calculated using the proposed AQM methods and the compared methods. Packet arrival is simulated based on a pre-determined probability value called "Alpha" that is in the range between 0-1. A random number generation is used to generate a random number to be compared with alpha. If the number generated is below alpha then the packet is arrived to the buffer, else no packet is arrived. The properties of random number besides it is random, it also generates a distributed number. When Alpha is 0.5 then for a 100 slots, 50 slots will allow packet arrival. Packet departure is simulated based on a pre-determined value called "Beta" and is happened in a similar way as the packet arrival and based on random number generation. Congestion Control checks the statuses and decides whether to drop the packet or to queue it. In the same time slot, a packet maybe departed. This process is repeated for each time slot. This process is illustrated in Figure 4.1.

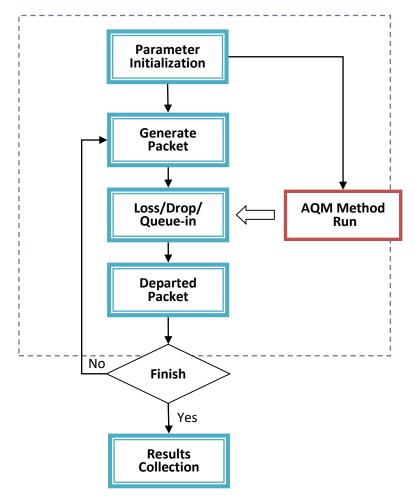


Fig 4.1: Flowchart of the Experiments

4.4 Results

The results of the proposed and compared methods are reported in this section.

Each group of the proposed methods is compared with RED and ERED using the performance measures mentioned earlier. The results are collected under different ALPHA's values to provide a variety of tests.

4.4.1 Queue Length-based Proposed Methods

In this sub-section, the three proposed methods, which was developed using queue-length indicator, as discussed in Chapter Three, are evaluated and compared with RED and ERED.

Table 4.2 shows the average delay for the proposed queue-length based methods and compared methods. Table 4.2 is plotted in Figure 4.2. As demonstrated in Figure 4.2, delay for RED, ERED and the proposed sub-methods at arrival probability 0.4 and below is almost identical. Above this value, ERED getting more value for delay compared to RED and the proposed sub-methods. As noted, for the values 0.3 and 0.4, the proposed sub-method-2 and the proposed sub-method-3 have achieved the same best delay, compared to the proposed sub-method-1, RED and ERED. However, at the value 0.5 and above the proposed sub-method-2 achieved the best delay compare with the rest.

Table 4.2: Delay Comparison for the Proposed Queue-based Methods

Alpha	RED	ERED	Sub-Method-1	Sub-Method-2	Sub-Method-3
0.3	260	169	41243	12375	4675
0.4	850	722	55617	2620	2620
0.5	3181	28653	13444	34573	32387
0.6	128606	124150	117175	126375	125052
0.7	237797	232512	234788	269303	269303
0.8	355110	340017	352669	360288	359901
0.9	473148	378621	472833	478906	478906
0.99	582929	518954	584232	587682	587682

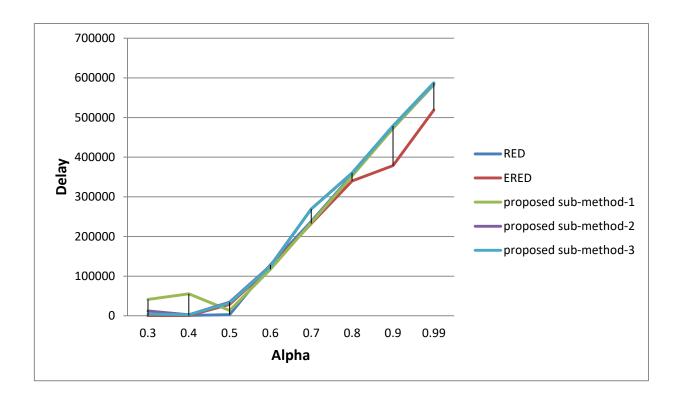


Fig 4.2: Delay Comparison for the Proposed Queue-based Methods

Drop Comparison

Table 4.3 shows the total dropping for the proposed queue-length based methods and compared methods. Table 4.3 is plotted in Figure 4.3. As demonstrated in Figure 4.3, the drop for RED, ERED and the proposed sub-methods2,3, at arrival probability 0.4 and below is almost identical. Above this value, ERED and the proposed method-1 has achieved the best results. However, the good result of the proposed method-1 was maintained up to the value of 0.6.

Table 4.3: Drop Comparison for the Proposed Queue-based Methods

Alpha	RED	ERED	Sub-method-1	Sub-method-2	Sub-method-3
0.3	0	0	0	75	75
0.4	0	82	7	2620	2620
0.5	3181	28653	13444	34573	32387
0.6	128606	114150	117175	126375	125052
0.7	237797	202512	234788	244303	240057
0.8	355110	240017	352669	360288	359901
0.9	473148	278621	472833	478906	478906
0.99	582929	318954	584232	587682	587682

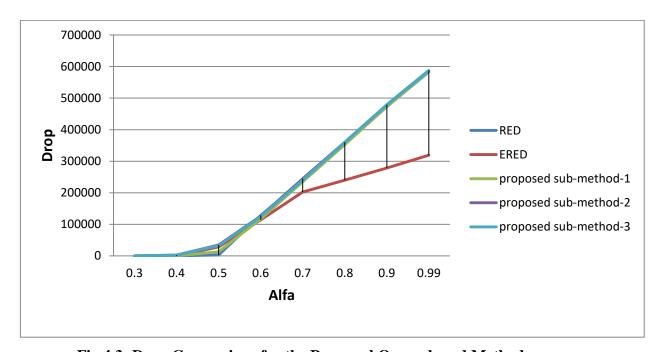


Fig 4.3: Drop Comparison for the Proposed Queue-based Methods

Loss Comparison

Table 4.4 shows the total loss for the proposed queue-length based methods and compared methods. Table 4.4 is plotted in Figure 4.4. As demonstrated in Figure 4.4, RED, ERED and the proposed sub-methods all have an equal loss at the value 0.3. At value of 0.4, the proposed sub-method-2 and sub-method-3 achieve the best loss and maintain it.

Alpha **RED ERED Sub-method-1** Sub-method-2 Sub-method-3 0.3 0.4 0.5 0.6 0.7 8.0 0.9 0.99

Table 4.4: Loss Comparison for the Proposed Queue-based Methods

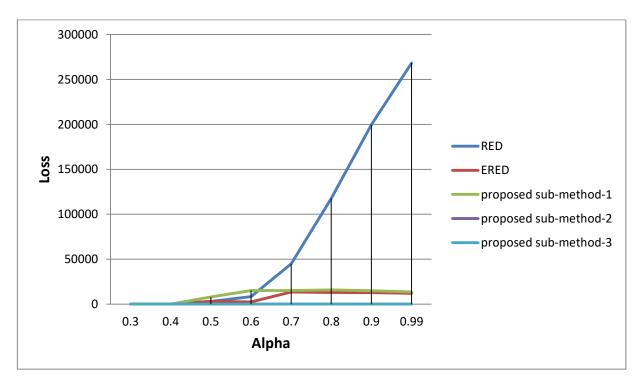


Fig 4.4: Loss Comparison for the Proposed Queue-based Methods

Table 4.5 shows the total drop and loss for the proposed queue-length based methods and compared methods. Table 4.5 is plotted in Figure 4.5. As demonstrated in Figure 4.5, RED, ERED and the proposed sub-method-1 at the value 0.3 had recorded the best drop and loss comparing to the proposed sub-method-2 and the proposed sub-method-3. Then, the best drop and loss is recorded for the proposed sub-method-2 and the proposed sub-method-3. However, the result clearly changed at the value 0.5 onward for sub-method-2 and maintained the best for the proposed sub-method-3. The Proposed sub-method 3 has achieved and maintained the best drop and loss value till the end.

Table 4.5: Drop and Loss Comparison for the Proposed Queue-based Methods

Alpha	RED	ERED	Sub-method-1	Sub-method-2	Sub-method-3
0.3	0	0	0	21	22
0.4	13	2	14	55	55
0.5	6426	4785	3598	5735	5382
0.6	19009	16169	18248	17414	17317
0.7	29967	28584	30043	29083	28576
0.8	38402	37494	38588	37497	37494
0.9	45164	44388	45238	44388	44388
0.99	62142	61822	50222	49482	49482

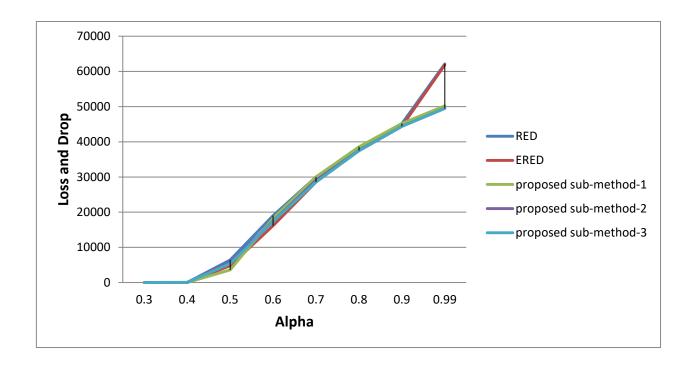


Fig 4.5: Drop and Loss Comparison for the Proposed Queue-based Methods

Overall, sub-proposed-3 and sub-method-2 have shown to give the best results compared with the rest. ERED has also satisfactory results by the means of all the utilized measures.

4.4.2 Delay-based Proposed Methods

In this sub-section, the three proposed methods, which was developed using delay indicator, as discussed in Chapter Three, are evaluated and compared with RED and ERED.

Table 4.6 shows delay for the proposed delay-based methods and compared methods. Table 4.6 is plotted in Figure 4.6. As demonstrated in Figure 4.6, at value of 0.3, RED and ERED and the proposed sub-method-4 and proposed sub-method-6 give the same delay value. The proposed sub-method-5 clearly gives a higher delay value. Those results are not maintained at the value of 0.6 and above, as the proposed sub-method-5 provides the best results.

Alpha	RED	ERED	Sub-method-4	Sub-method-5	Sub-method-6
0.3	0	0	0	0	0
0.4	0	82	60	60	60
0.5	3181	28653	5334	2384	2290
0.6	128606	114150	13262	7476	7474
0.7	237797	202512	14416	10766	11104
0.8	355110	240017	12447	8845	8822
0.9	473148	278621	11680	7281	6842
0.99	582929	318954	10780	6175	6218

Table 4.6: Delay Comparison for the Proposed Delay-based Methods

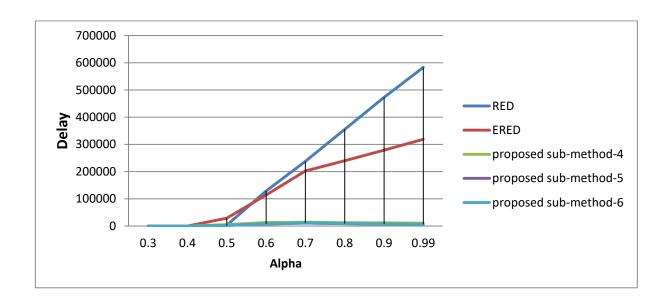


Fig 4.6: Delay Comparison for the Proposed Delay-based Methods

Drop Comparison

Table 4.7 shows the drop for the proposed delay-based methods and compared methods. Table 4.7 is plotted in Figure 4.7. As demonstrated in Figure 4.7, RED, ERED and Proposed sub-methods have identical results at the value of 0.3. However, at the value 0.4, sub-method-5 and the proposed sub-method-6 provide the best dropping value compared with the rest, but not in a major manner. At value of 0.5, sub-method-6 gives the best results marginally. For the rest of the results, sub-method-5 gives the best results.

Alpha	RED	ERED	Sub-method-4	Sub-method-5	Sub-method-6
0.3	0	0	0	0	0
0.4	0	82	0	1	1
0.5	3181	28653	26536	32544	32498
0.6	128606	114150	120296	123870	124180
0.7	237797	202512	235261	235632	234681
0.8	355110	240017	354204	353406	354385
0.9	473148	278621	474731	473001	471612
0.99	582929	318954	583610	582540	582252

Table 4.7: Drop Comparison for the Proposed Delay-based Methods

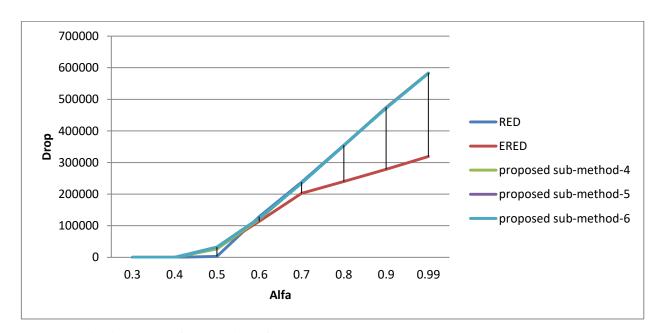


Fig 4.7: Drop Comparison for the Proposed Delay-based Methods

Loss Comparison

Table 4.8 shows the loss for the proposed delay-based methods and compared methods. Table 4.8 is plotted in Figure 4.8. As demonstrated in Figure 4.8, RED, ERED and all the proposed sub-methods provides the same loss value, at the value of 0.3. At the value of 0.4, ERED provides the best loss but not with a significant margin, compares with the rest. At the value of 0.5, ERED produces the best results. The proposed sub-method-5 and the proposed sub-method-6 provide and maintain the best loss value compare to the rest.

Table 4.8: Loss Comparison for the Proposed Delay-based Methods

Alpha	RED	ERED	Sub-method-4	Sub-method-5	Sub-method-6
0.3	0	0	0	0	0
0.4	18	60	60	60	60
0.5	122	3188	5334	2384	2290
0.6	2592	8381	13262	7476	7474
0.7	44842	13484	14416	10766	11104
0.8	117555	13087	12447	8845	8822
0.9	199617	12828	11680	7281	6842
0.99	268144	12103	10780	6175	6218

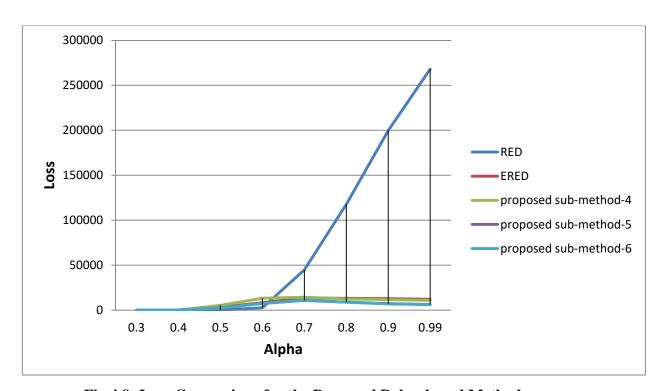


Fig 4.8: Loss Comparison for the Proposed Delay-based Methods

Table 4.9 shows the drop and loss aggregation for the proposed delay-based methods and compared methods. Table 4.9 is plotted in Figure 4.9. As demonstrated in Figure 4.9, there is no margin differences in the results of all the compared and the proposed methods. Overall, drop and loss for RED, ERED and the proposed sub-methods at the value 0.3 are identical. At the value of 0.4, RED and the proposed sub-methods provides a closely similar drop and loss value. At the other values, ERED and sub-methods-6 give the best result.

Table 4.9: Drop and Loss Comparison for the Proposed Delay-based Methods

Alpha	RED	ERED	Sub-method-4	Sub-method-5	Sub-method-6
0.3	0	0	0	0	0
0.4	32	13	12	12	12
0.5	6426	4785	5328	5771	575
0.6	19009	16169	1859	18229	18272
0.7	29967	28584	29513	29237	29262
0.8	38402	37494	38181	37702	37857
0.9	45164	44388	44976	44429	44449
0.99	50114	40482	49983	49469	49549

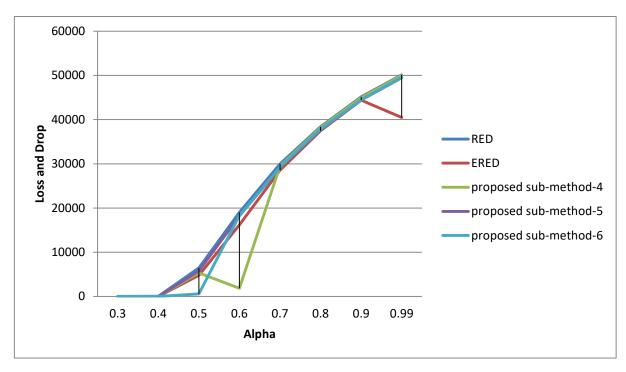


Fig 4.9: Drop and Loss Comparison for the Proposed Delay-based Methods

Overall, ERED has shown to give the best results compared with the rest. The proposed sub-method-6 and RED have also satisfactory results by the means of all the utilized measures.

4.4.3 Load-based Proposed Methods

In this sub-section, the three proposed methods, which was developed using load indicator, as discussed in Chapter Three, are evaluated and compared with RED and ERED.

Delay Comparison

Table 4.10 shows delay for the proposed load-based methods and compared methods. Table 4.10 is plotted in Figure 4.10. As demonstrated in Figure 4.10, at values of 0.3 and 0.4, RED, ERED and the proposed sub-methods have the same delay value. At the value of 0.5, the results of all the methods are also almost similar with slightly

better for ERED. At the value 0.6 and above the proposed sub-method-7 and ERED give the best delay value compare with the rest.

Table 4.10: Delay Comparison for the Proposed Load-based Methods

Alpha	RED	ERED	Sub-method-7	Sub-method-8	Sub-method-9
0.3	350773	350773	350773	30773	350773
0.4	584475	585111	585122	585119	585122
0.5	1240181	1389808	1780169	1054485	1834025
0.6	2054205	1821025	1956409	2022601	2394007
0.7	2090896	1926622	1957856	2332519	2634898
0.8	2040384	1949447	1948204	2542172	2682462
0.9	2035898	1902153	1939444	2651028	2730972
0.99	2007244	1953334	1931982	2690246	2266752

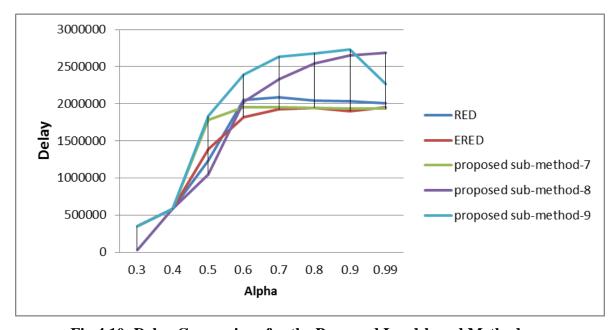


Fig 4.10: Delay Comparison for the Proposed Load-based Methods

Drop Comparison

Table 4.11 shows the drop values for the proposed load-based methods and compared methods. Table 4.11 is plotted in Figure 4.11. As demonstrated in Figure 4.11, the drop for RED, ERED and the proposed sub-methods are similar in arrival probability of 0.3, which will be not maintained at the value 0.4. At the value of 0.6 to the value 0.9, ERED provide the best drop value compared with the rest. The proposed sub-method-7 gives satisfactory results as well.

Sub-method-7 Alpha **RED ERED Sub-method-8** Sub-method-9 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.99

Table 4.11: Drop Comparison for the Proposed Load-based Methods

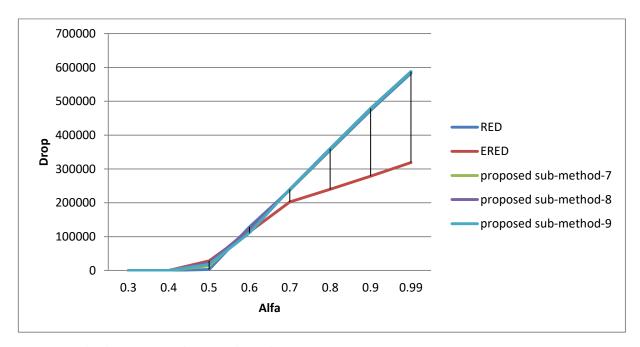


Fig 4.11: Drop Comparison for the Proposed Load-based Methods

Loss Comparison

Table 4.12 shows the loss values for the proposed load-based methods and compared methods. Table 4.12 is plotted in Figure 4.12. As demonstrated in Figure 4.12, RED, ERED and the proposed sub-methods, at the value 0.4, give similar loss and that will be changed at the value 0.5, as only ERED will provide the best loss among all. However from the value 0.7 and above, the proposed sub-method-8 and sub-method-9 provide the best loss.

Table 4.12: Loss Comparison for the Proposed Load-based Methods

Alpha	RED	ERED	Sub-method-7	Sub-method-8	Sub-method-9
0.3	0	0	0	0	0
0.4	18	60	60	60	60
0.5	122	3188	7658	2005	3435
0.6	2592	8381	10660	1444	9
0.7	44842	13484	9341	1	0
0.8	117555	13087	7293	0	0
0.9	199617	12828	5998	0	0
0.99	268144	12103	5093	0	0

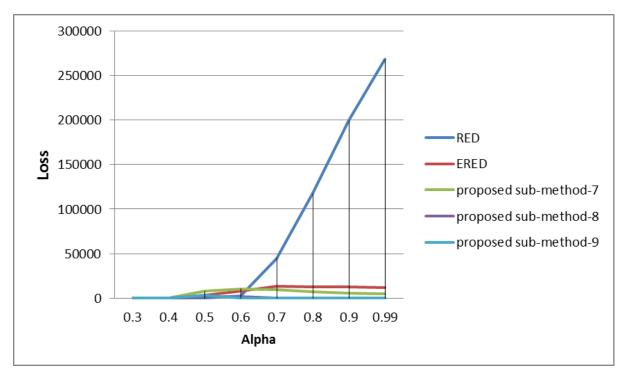


Fig 4.12: Loss Comparison for the Proposed Load-based Methods

Table 4.13 shows the aggregated loss and drop values for the proposed load-based methods and compared methods. Table 4.13 is plotted in Figure 4.13. As demonstrated in Figure 4.13, RED, ERED and the proposed sub-methods, at the value 0.4, give similar loss and that will be changed at the value 0.5, as only ERED will provide the best loss among all. However from the value 0.7 and above, the proposed sub-method-8 and sub-method-9 provide the best loss.

49482

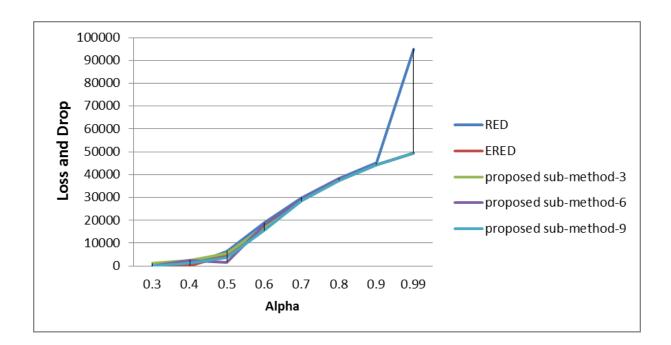
Alpha	RED	ERED	Sub-method-7	Sub-method-8	Sub-method-9
0.3	0	0	125	0	0
0.4	325	125	2355	254	125
0.5	6426	4785	5382	1575	3665
0.6	19009	16169	17317	18272	15581
0.7	29967	28584	28576	29262	28583
0.8	38402	37494	37494	37857	37494
0.9	45164	44388	44388	44449	44388

0.99

95014

49482

Table 4.13: Drop and Loss Comparison for the Proposed Load-based Methods



49482

49549

Fig 4.13: Drop and Loss Comparison for the Proposed Load-based Methods

Overall, ERED has shown to give the best results compared with the rest. The proposed sub-method-9 and sub-method-3 have also satisfactory results by the means of all the utilized measures.

4.4.4 Best of the Proposed Methods

In this sub-section, the best proposed sub-method from each category, queu-length based, delay-based and load-based are compared in order to give a final conclusion about the proposed work.

Delay Comparison

Table 4.14 shows the delay values for the best methods and compared methods. Table 4.14 is plotted in Figure 4.14. As noted the sub-method-2 and sub-method-6 outperformed the compared methods.

Table 4.14: Delay Comparison for the Best of the Proposed Methods

Alpha	RED	ERED	Sub-method-2	Sub-method-6	Sub-method-9
0.3	350773	340773	349861	350773	350773
0.4	585111	584475	536871	585103	585122
0.5	1089808	1040181	1085234	1041137	1434025
0.6	1818025	1554205	1305974	1362951	1394007
0.7	1626622	1590896	1536583	1584558	1634898
0.8	1649447	1540384	1660045	1696903	1682462
0.9	2219553	1735898	1773079	1797766	1730972
0.99	1953334	1807244	1268166	1294301	1766752

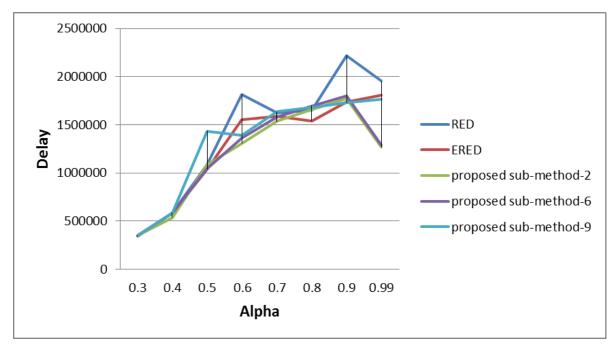


Fig 4.14: Delay Comparison for the Best of the Proposed Methods

Drop Comparison

Table 4.15 shows the drop values for the best methods and compared methods.

Table 4.15 is plotted in Figure 4.15. As noted, ERED outperformed the other methods.

Alpha	RED	ERED	sub-method-1	sub-method-6	sub-method-7
0.3	0	0	0	0	0
0.4	0	82	7	1	0
0.5	3181	28653	13444	32498	14141
0.6	128606	114150	117175	124180	118853
0.7	237797	202512	234788	234681	238302
0.8	355110	240017	352669	354385	358842
0.9	473148	278621	472833	471612	477153
0.99	582929	318954	584232	582252	587209

Table 4.15: Drop Comparison for the Best of the Proposed Methods

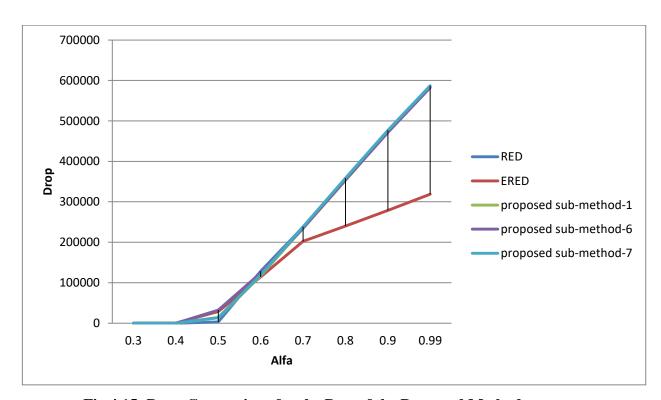


Fig 4.15: Drop Comparison for the Best of the Proposed Methods

Loss Comparison

Table 4.16 shows the loss values for the best methods and compared methods. Table 4.16 is plotted in Figure 4.16. As noted the sub-method-2 and sub-method-9 outperformed the compared methods.

Alpha	RED	ERED	Sub-method-2	Sub-method-6	Sub-method-9
0.3	0	0	0	0	0
0.4	18	60	0	60	60
0.5	122	3188	0	2290	2435
0.6	2592	1381	0	2474	9
0.7	14842	13484	0	1104	0
0.8	13555	13087	0	8822	0
0.9	13617	12828	0	6842	0
0.99	18144	12103	0	6218	0

Table 4.16: Loss Comparison for the Best of the Proposed Methods

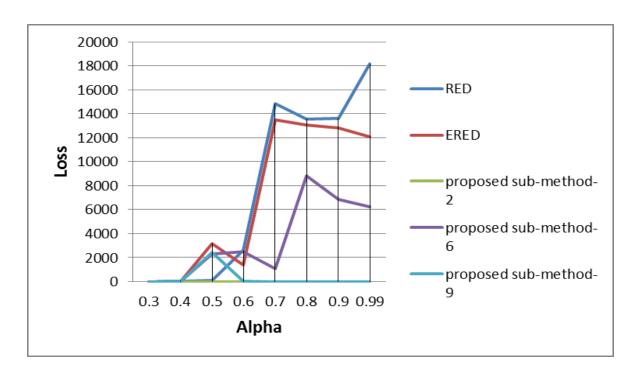


Fig 4.16: Loss Comparison for the Best of the Proposed Methods

Table 4.17 shows the aggregated drop and loss values for the best methods and compared methods. Table 4.17 is plotted in Figure 4.17. As noted, ERED outperformed the other methods, and the best drop and loss is sub-method-7.

Table 4.17: Drop and Loss Comparison for the Best of the Proposed Methods

Alpha	RED	ERED	Sub-method-2	Sub-method-6	Sub-method-9
0.3	0	0	0	0	0
0.4	130	200	130	130	130
0.5	6426	3585	3584	4129	3665
0.6	19009	15169	18145	17093	15581
0.7	29967	20584	29678	22843	28583
0.8	38402	37424	38284	37494	37494
0.9	45164	44311	44364	44388	44388
0.99	45014	43482	44991	49482	49482

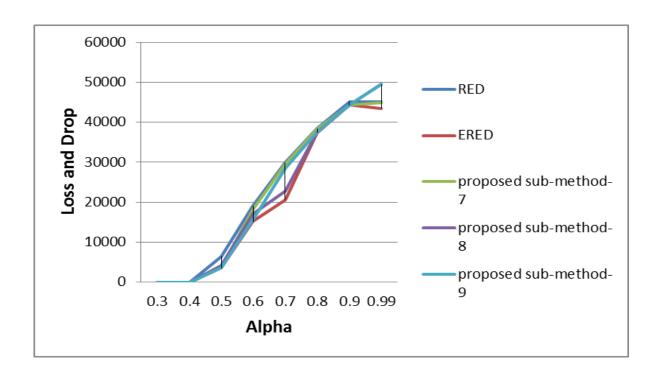


Fig 4.17: Drop and Loss Comparison for the Best of the Proposed Methods

4.5 Summary

This chapter evaluates and compares the proposed methods. In summary of the findings, the one that provides the best delay is sub-method 6 and best loss is the proposed sub-method 9. However, ERED provides the best results according to some measurements. Overall, the proposed sub-method 6 is the provider of the best drop. Loss and Drop rate as noted are best achieved by the proposed sub-method 7. Subsequently, each of them can be used according to the type of the network.

CHAPTER FIVE

Conclusions and Future Works

5.1 Conclusion

This thesis presents an approach for modifying and replacing the existing RED's indicator with new indicators (Queue length, load rate and Delay). Every time we do such replacement, we obtained a new method, which provided us with nine different proposed methods. All these proposed methods were tested and studied in order to find out which one of these nine is going to be recognized as the best method.

We compared the proposed methods with RED and ERED. In summary of the findings, the one that provides the best delay and best loss is the proposed sub-method-2. However, ERED provides the best results according to some measurements. Overall, the proposed sub-method-6 is the provider of the best drop. Loss and Drop rate as noted are best achieved by the proposed sub-method-2. Subsequently, each of them can be used according to the type of the network.

5.2 Future Work

In the future, We will experiment some future works:

- We can Use different algorithm rather than Modified Random Early Detection (RED)
 algorithm with various indicators and implement, test and evaluates the modified that
 algorithm.
- we can Use different indicators with Modified Random Early Detection (RED) and other algorithms. Based on the previous solutions and implement, test and evaluates the modified that algorithm.
- 3. we can propose algorithm that deal with two or multi-buffers in the router to prevent congestion before it happens and reduce congestion.

References

References

- Abbasov, B., & Korukoglu, S. (2009). Effective RED: An algorithm to improve RED's performance by reducing packet loss rate. *Journal of Network and Computer Applications*, 32(3), 703-709.
- Alemu, T. (2004). Evaluation des Performances des Mécanismes de Qualité de Service dans l'Internet. These de doctorat, Université de Montpellier II.
- Alfa, A. S. (2010). Queueing theory for telecommunications: discrete time modelling of a single node system. Springer Science & Business Media.
- Baker, F., & Fairhurst, G. (2015). *IETF Recommendations Regarding Active Queue Management* (No. RFC 7567).
- Balkas, Y. (2002). DELAY-BOUNDED RATE ADAPTIVE SHAPER FOR TCP TRAFFIC IN DIFFSERV INTERNET (Doctoral dissertation, bilkent university).
- Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., ... & Peterson, L. (1998). Recommendations on queue management and congestion avoidance in the Internet (No. RFC 2309).
- Brazio, J., Tran-Gia, P., Akar, N., Beben, A., Burakowski, W., Fiedler, M., ... & Wittevrongel, S. (Eds.). (2006). *Analysis and Design of Advanced Multiservice Networks Supporting Mobility, Multimedia, and Internetworking: COST Action* 279 Final Report. Springer Science & Business Media.
- Chen, J., Hu, C., &Ji, Z. (2010).Self-tuning random early detection algorithm to improve performance of network transmission.Mathematical Problems in Engineering, 2011.
- Chua, K. C., Gurusamy, M., Liu, Y., & Phung, M. H. (2007). *Quality of service in optical burst switched networks*. Springer Science & Business Media.

- Dhodapkar, A. S., & Smith, J. E. (2002). Managing multi-configuration hardware via dynamic working set analysis. In *Computer Architecture*, 2002. *Proceedings*.
 29th Annual International Symposium on (pp. 233-244). IEEEFeng, W. C., Shin, K. G., Kandlur, D. D., & Saha, D. (2002). The BLUE active queue management algorithms. *IEEE/ACM Transactions on Networking (ToN)*, 10(4), 513-528.
- Feng, W. C., Kandlur, D., Saha, D., & Shin, K. (1999). BLUE: A new class of active queue management algorithms. *Ann Arbor*, *1001*, 48105.
- Feng, W. C., Kandlur, D., Saha, D., & Shin, K. G. (2001, January). Blue: An alternative approach to active queue management. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video* (pp. 41-50). ACM.
- Feng, W. C., Shin, K. G., Kandlur, D. D., & Saha, D. (2002). The BLUE active queue management algorithms. *IEEE/ACM Transactions on Networking* (*ToN*), *10*(4), 513-528.
- Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, *1*(4), 397-413.
- Floyd, S., Handley, M., Padhye, J., & Widmer, J. (2000). Equation-based congestion control for unicast applications. *ACM SIGCOMM Computer Communication Review*, 30(4), 43-56.
- Freed, M., & Amara, S. K. (2006). *U.S. Patent No. 6,996,062*. Washington, DC: U.S. Patent and Trademark Office.
- Homg, M. F., Lee, W. T., Lee, K. R., & Kuo, Y. H. (2001). An adaptive approach to weighted fair queue with QoS enhanced on IP network. InTENCON 2001.
 Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology (Vol. 1, pp. 181-186). IEEE.

- Hu, N., Ren, L., & Chang, J. (2001). Evaluation of queue management algorithms. *Course Project report for Computer Networks*, 1-14.
- Janevski, T. (2003). *Traffic analysis and design of wireless IP networks*. Artech House.
- Lee, K. M., Yang, J. H., &Suh, B. S. (2008). Congestion Control of Active Queue
 Management Routers Based on LQ-Servo Control. *Engineering Letters*, 16(3), 332-338.
- May, M., Bolot, J., Diot, C., & Lyles, B. (1999). Reasons not to deploy RED.In *Quality of Service*, 1999.IWQoS'99. 1999 Seventh International Workshop on (pp. 260-262). IEEE.
- Misra, S., Oommen, B. J., Yanamandra, S., &Obaidat, M. S. (2010). Random early detection for congestion avoidance in wired networks: a discretized pursuit learning-automata-like solution. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 40(1), 66-76.
- Mohamed, M. H. E. (2010). Some Active Queue Management Methods for Controlling Packet Queueing Delay. Design and Performance Evaluation of Some New Versions of Active Queue Management Schemes for Controlling Packet Queueing Delay in a Buffer to Satisfy Quality of Service Requirements for Real-time Multimedia Applications (Doctoral dissertation, University of Bradford).
- Thiruchelvi, G., & Raja, J. (2008). A survey on active queue management mechanisms. *International Journal of Computer Science and Network Security*, 8(12), 130-145.
- Odom, W., & Cavanaugh, M. J. (2004). Cisco QOS Exam Certification Guide (IP Telephony Self-Study). Pearson Education.

- Pan, R., Prabhakar, B., & Psounis, K. (2000). CHOKe-a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM* 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 2, pp. 942-951). IEEE.
- Pan, R., Prabhakar, B., &Psounis, K. (2000).CHOKe-a stateless active queue management scheme for approximating fair bandwidth allocation. In INFOCOM 2000.Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.Proceedings.IEEE (Vol. 2, pp. 942-951).IEEE.
- Rastogi, S., & Srivastava, S. (2014). Comparison Analysis of Different Queuing Mechanisms Droptail, RED and NLRED in Dumb-bell Topology. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(4).
- Ren, F., He, T., Das, S. K., & Lin, C. (2011). Traffic-aware dynamic routing to alleviate congestion in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(9), 1585-1599.
- Rosolen, V., Bonaventure, O., & Leduc, G. (1999). A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic. *ACM SIGCOMM Computer Communication Review*, 29(3), 23-43.
- Silberschatz, A., Ozden, B., Bruno, J., & Saran, H. (2003). *U.S. Patent No.* 6,556,578. Washington, DC: U.S. Patent and Trademark Office.
- Stoica, I., Shenker, S., & Zhang, H. (1998). Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks (Vol. 28, No. 4, pp. 118-130). ACM.

- Thiruchelvi, G., & Raja, J. (2008). A survey on active queue management mechanisms. *International Journal of Computer Science and Network Security*, 8(12), 130-145.
- Wang, Y. C., Jiang, J. A., & Chu, R. G. (2004, August). Drop behaviour of random early detection with discrete-time batch Markovian arrival process.

 In *Communications, IEE Proceedings* (Vol. 151, No. 4, pp. 329-336).IET.
- Wang, B., Kasthurirangan, B., & Xu, J. (2005). Subsidized RED: an active queue management mechanism for short-lived flows. *Computer communications*, 28(5), 540-549.
- Wu-changFeng, Kang Shin, DilipKandlur, DebanjanSaha, The BlueActive Queue
 Management Algorithms, IEEE/ACM Transactions onNetworking, Vol. 10, No.
 4, August 2002.
- Wurtzler, M. (2002). *Analysis and simulation of weighted random early detection* (WRED) queues (Doctoral dissertation, University of Kansas).
- Xie, X. (2008). A review of recent advances in surface defect detection using texture analysis techniques. *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, 7(3).

- Xu, K., Gerla, M., Qi, L., & Shu, Y. (2005). TCP unfairness in ad hoc wireless networks and a neighborhood RED solution. Wireless Networks, 11(4), 383-399.
- Zhang, C., Yin, J., Cai, Z., & Chen, W. (2010). RRED: robust RED algorithm to counter low-rate denial-of-service attacks. *IEEE Communications Letters*, *14*(5), 489-491.